MÄLARDALEN UNIVERSITY
SWEDEN

SCANIA

Building a Safety Case in Compliance with ISO 26262 for Fuel Level Estimation and Display System

Master Thesis in Intelligent Embedded Systems
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden

Author
Raghad Dardar

Supervisors:
Barbara Gallina, Mälardalen University
Mattias Nyberg, Scania CV AB

Examiner:
Kristina Lundqvist, Mälardalen University

September, 2013

Abstract

Nowadays, road vehicles, including trucks, are characterized by an increased complexity due to a greater variety of software, and a greater number of sensors and actuators. As a consequence, there is an increased risk in terms of software or hardware failures that could lead to unacceptable hazards. Thus safety, more precisely functional safety, is a crucial property that must be ensured to avoid or mitigate these potential unacceptable hazards. In the automotive domain, recently (November 2011), the ISO-26262 safety standard has been introduced to provide appropriate requirements and processes. More specifically, the standard defines the system development process that must be carried out to achieve a system that can be considered acceptably safe. To be released on the market, systems must be certified, proofs that the systems are acceptably safe must be provided in terms of a structured argument, known as safety case, which inter-relates evidence and claims. Certification authorities are in charge of evaluating the validity of such safety cases. In the automotive domain, certification and compliance with the standard ISO-26262 is becoming mandatory. By now, trucks do not have to be compliant with the standard. However, it is likely that by 2016 they will have to. Scania is one of the leading companies in trucks development. To be ready by 2016, Scania is interested in investigating ISO-26262 as well as safety case provision. Thus this thesis focuses on the provision of a safety case in the context of ISO-26262 for Fuel Level Estimation and Display System (FLEDS), which is one of the safety-critical systems in Scania.

Contents

List of Tables

List of Figures

ABBREVIATIONS

AE	Allocation Element
AER	Allocation Element Requirement
APPL	Application Layer
ASIL	Automotive Safety Integrity Level
Cl	Clause
COO	Coordinator
EMS	Engine Management System
FAA	Federal Aviation Administration
FSR	Functional Safety Requirement
FTA	Fault Tree Analysis
FMEA	Failure Mode and Effect Analysis
FLEDS	Fuel Level Estimation and Display System
HW	Hardware
HAZOP	Hazard and Operality Study
ICL	Instrument Cluster
KF	Kalman Filter
MIDD	Middle Layer
NGC	New Generation Scania
Pt	Part
PHA	Preliminary Hazard Analysis
RTDB	Real Time DataBase
SG	Safety Goal
SW	Software
TSR	Technical Safety requirement
UF	User Function
UFR	User Function Requirement

1. INTRODUCTION

This chapter presents a brief introduction about the safety case, ISO 26262 and the system under study. Moreover, context and motivation and the organization of this thesis are presented further in the next sections.

The need for safety cases that are used to argue and demonstrate the safety of the system has appeared. Safety case is a contextualized structured argument that links evidence to claims to show that the system is acceptably safe. There are twdo types of arguments that are used to construct the safety case; product and process-based arguments. Product-based argument is an argument that is used to show that the behaviour of the system is safe by using evidence that is related to the product's behaviour, whereas process-based argument is the one that is used to show that the process followed during the development life cycle is trustworthy. More details about these arguments are presented further in chapter 2.
With the introduction of safety standards such as ISO 26262, there was a change in the approach being followed in safety management as these standards provide the best practices that should be followed during the development, construction and operation of safety critical system.
Quoting from the standard: ISO-26262 is intended to be applied to safety-related systems that include one or more electrical and/or electronic (E/E) systems and that are installed in series production passenger cars with a maximum gross vehicle mass up to 3 500 kg. This thesis is about building a safety case for Fuel Level Estimation and Display System (FLEDS), which is one of the safety-critical systems in Scania trucks. Indeed, a wrong behaviour (e.g. false fuel level) of such system could lead to hazardous events for the driver, such as engine stop and loss of power assisted steering. The focus of the thesis is to show how the different work products that are generated during the development life-cycle can be used to construct sub-arguments that are to be used to build the partial safety case. Moreover, the focus will be on collecting and providing (when missing) the evidence (product and process evidence) required for constructing the sub-arguments that are used to construct the safety case. In order to build a safety case in compliance with ISO 26262, mapping of what the system has to the standard is required. More detailed information about mapping is presented later in this thesis.

1.1 Context and motivation

Scania is interested in exploring ISO 26262 and the provision of the safety case. Therefore, the safety case is built in the context of ISO 26262. Only specific parts of ISO 26262 have been covered due to time limitation. This thesis will help Scania to address what is needed and missing in case Scania is interested in certification. Moreover, experiences from building a safety case in the context of ISO 26262 in an industrial setting are not so many and therefore such experiences are valuable and presented in the end of this thesis. Moreover, throughout this thesis, we will try to address the following questions:

- What does it mean to build a safety case in compliance with ISO 26262?
- What does it mean to have a clear separation between process and product-based arguments?
- What kind of work-products are required to be in compliance to the standard?
- What are the estimated effort for complete conformity with ISO 26262?

1.2 Contributions

The author of this thesis has provided a partial safety case for an industrial setting [FLEDS] in the context of ISO 26262. A clear separation between product and process-based arguments have been maintained during this thesis as well. Moreover, a paper [25] presented at the 23rd International Symposium on Software Reliability (ISSRE) has steemed from this thesis.

1.3 Organization of the thesis

The rest of the report is organized as follows: In chapter 2, essential background necessary for this thesis as well as the related work are presented. In chapter 4, the solution methods used to achieve the results of this thesis are presented. In chapter 5, how we have collected or provided (when missing) the evidence needed to develop illustrative process and product-based arguments through mapping are explained. In chapter 6, the safety case of the system is presented. Moreover, lessons learned and general guidelines to facilitate the adoption of ISO 26262 and the creation of safety cases are presented in chapter 7. Conclusion remarks and future work are presented in chapter 8. Finally, Contributions of this thesis are presented in section 1.2

2. BACKGROUND and RELATED WORK

This chapter introduces the background information that are essential for this thesis. Section 2.1 introduces essential information about the concept of the *user function* in Scania. Consecutive sections introduce essential infromation about hazard analysis, ISO 26262 and safety case.

2.1 User Functions

In Scania, there's a possibility to select different driver experience functions based on customer demands. These functions are called user functions (UF). UF is a function that the user can perceive on the vehicle level [29], for example air condition function is a UF since the driver can control the adjustment of the air condition. User functions can be seen as high level customer requirements [29].

2.1.1 Implementation of User Functions

To be able to implement UFs, one or more ECUs are required for each function. Each ECU unit consists of specific hardware and software that are needed to implement the specific UF. Moreover, communication among ECUs is also essential in order to have a complete working system. The following subsections explain what is required to implement a UF.

Electronic control unit

The electronic control unit (ECU) is a vital part to do all the required operations to implement the UF. Each ECU consists of hardware and software in which the hardware of an ECU consists of sensors and actuators. ECUs are the main components that are used in the electrical system of Scania which is called Scania Electrical System Architecture for Modularization and Maintenance (SESAMM).

Controller Area Network

The communication among different ECUs is established through a controller area network (CAN). CAN communication in SESAMM is divided into three buses which are *red* CAN bus, *yellow* CAN bus, and *green* CAN bus. The distribution of ECUs to these buses is as follows:

- The ECUs of the drive-line system are connected on a separate bus, the red bus. This is done to protect the drive-line functions from other less important functions from causing a failure on the bus. The ECU systems of the red bus are considered to be of highest criticality. These systems have criticality 1 [8].
- The ECUs of the most important systems that are not part of the drive-line are connected on a separate bus, the yellow bus, for the same reasons as given for the red bus. These systems have criticality 2 [8].
- The rest of the ECU systems are placed on the green bus. These systems have criticality 3 [8].

Allocation Element

Allocation element (AE) is a piece of code that is responsible for realizing a UF. Each ECU unit contains several AEs where each UF is realized by one or more AE.

2.2 FUEL LEVEL ESTIMATION AND DISPLAY

In this section, FLEDS is presented in order to understand how the system works and what are the components needed to build the system. There are different variants of FLEDS and only one variant is considered in this thesis. Therefore, FLEDS variants are presented in this section. Moreover, allocation elements for FLEDS are presented since it's necessary to analyse AEs with respect to safety because AEs correspond to the code needed to implement the required functionality. The requirements of the systems are also necessary to be presented in this section in order to know later if these requirements have been met or not.
FLEDS has two major functions; Fuel level estimation and low fuel level warning. Truck driver needs to know how much fuel left in the tank and thus there's a need for a function that will estimate fuel level and display it to the driver. Therefore fuel level estimation function will be used. Sometimes the driver doesn't check the gauge frequently and thus there is a need for a mechanism that indicates that there's low fuel level in the tank. Thus low fuel level warning function will be used to indicate to the driver that

a refuel is needed. Fuel level is measured and presented to the driver in the instrument cluster. If the fuel level is below a predetermined level a warning is presented in the instrument cluster as well. UF18 is responsible for FLEDS. The system consists of three ECU (Electronic Control Unit) systems which are Engine Management System (EMS), Instrument Cluster (ICL), and Coordinator (COO). In order to be able to estimate and check the fuel level in the tank, a fuel level sensor is essential, in which it will be placed in the tank. A signal from parking brake switch (PBS) is necessary for refuel detection purposes. A battery is used to provide a power supply for ECU systems.

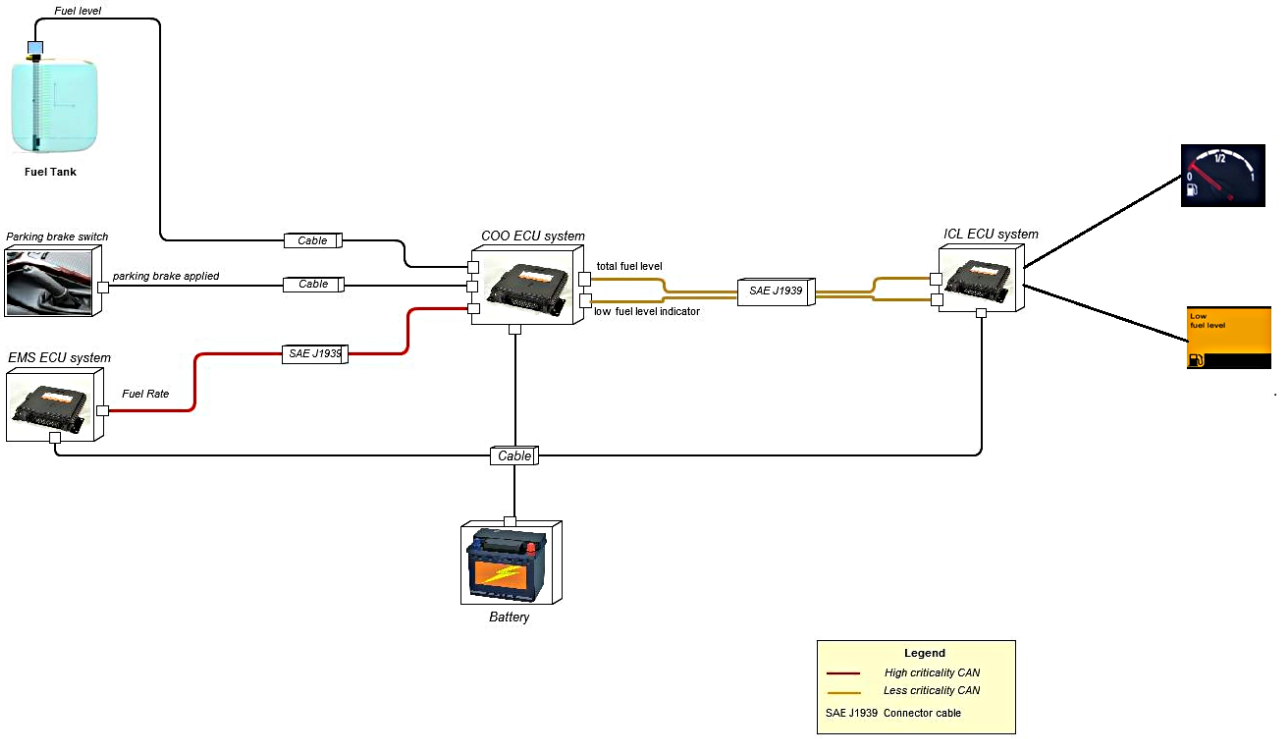


Figure 2.1: Components of Fuel Level Estimation and Display (FLEDS)

2.2.1 FLEDS variants

FLEDS will be used in two types of vehicles and with two different types of fuel (liquid and gas). Thus there are four variants for fuel level display system:

1. Truck with liquid fuel

2. Bus with liquid fuel

3. Truck with gas

4. Bus with gas

For the simplicity of this thesis, only one variant is considered. The considered variant is for truck with liquid fuel (variant 1) with one tank and one sensor. Moreover, there are different types of sensors and tank sizes. Therefore, only one sensor type and one tank size (left and right tank) is considered in this thesis. For the first variant of fuel level display system, the fuel level is estimated by COO.

Table 2.1: Configuration parameters for variant 1 of FLEDS

Variant 1: Truck with Liquid Fuel	
Configuration Parameters	Value
fuelLevelSensorParam	15 (Wema-general: short)
fuelTankSizeLeft	13 (450 litre volume: General)
fuelTankSizeRight	Same as for fuelTankSizeLeft
fuelLevelTotalParam	10 (Total fuel level calculated by COO)

2.2.2 FLEDS Allocation Elements

Two allocation elements have been used for the realization of UF18 (Fuel Level Estimation and Display): AE 201 and AE 202. AE 201 and AE 202 are allocated in COO.AE 201 handles the fuel level estimation part. The following steps are carried out for fuel level estimation:

1. The fuel level sensor is connected to COO and the level of the fuel is read as a voltage value [27].

2. The voltage value is transformed into the corresponding volume in percentage of the total volume. The percentage value is used together with the total fuel capacity of the tank in litres to calculate the current fuel volume [27].

3. A kalman filter algorithm is used to estimate the fuel level. The last fuel volume estimate is used together with the current volume calculated in step 2 and the fuel consumption from the engine to calculate a new estimate [27].

4. The estimated value from step 3 is transformed into a corresponding percentage value and sent to ICL for display of the current fuel level and to AE 202 for low fuel level indication [28].

AE 202 handles low fuel level warning. Information about the fuel level and tank capacity on the vehicle is used to activate a warning if the fuel level drops below a predefined level. The warning is kept even if the fuel level

Table 2.2: UFRs for FLEDS

UFR18	
UFR ID	Description
UFR18_1	The indicated fuel level shall not deviate more than ±5% from the actual volume in the fuel tank.
UFR18_4	The low fuel level warning shall warn one time when the estimated fuel level reaches below a limit of the measurable volume in the tank. The limit should be 10% for tank sizes below 900 litres and 7% for larger tanks.
UFR18_8	The estimated fuel level shall be shown immediately when the electrical system is switched on.

for some reason increases again after the warning is set. Unusual driving environments, such as steep hills and tough terrain, can affect the fuel level to increase a short period even if no fuel is filled. However if the fuel level increases a lot, caused by for example a refill, the warning is unset again [28].

2.2.3 FLEDS Requirements

There are two types of requirements for FLEDS: User Function Requirements (UFRs) and Allocation Element Requirements (AERs). UFRs are used to describe the high level requirements for UF18 whereas AERs describe how to implement UFRs. UFRs and AERs that have been covered in this thesis are presented below.

Table 2.3: AER 201 for FLEDS

AER 201	
AER ID	Description
AER_201_9	If the truck has one fuel level sensor connected, its voltage value should be transformed into a corresponding volume value in litres. The volume value should be low pass filtered before set to fuelLevel signal (ys).
AER_201_11	totalFuelLevel should be the output of a filter that includes information from both fuelLevel and fuelRate to achieve a stable signal. The filter should be implemented with a Kalman algorithm as given in equation $x_{start} = \begin{cases} y_s(t), & \lvert y_s - \hat{x}_{old} \rvert > 0.1X_{tot} \text{ or } y_s > 0.9X_{tot} \\ \hat{x}_{old}, & otherwise \end{cases}$ $\hat{x}(t+T_s) = \begin{cases} x_{start}, & during\ start-up \\ \hat{x}(t) - T_s u(t) + K\big(y_s(t) - \hat{x}(t)\big), & other \end{cases}$
AER_201_12	The start-up state for the totalFuelLevel estimated should be the state saved from last shutdown if the stored value and fuelLevel doesn't differ with more than 10% of the total volume or if fuelLevel is above 90%2 of the useable tank capacity.
AER_201_13	If a refill of the tank is done while the ECU is on it should be detected by the algorithm if the sensor(s) indicates a 30% increase compared to the estimated volume. The increase should be held at least 5 seconds so that sloshing is ignored.
AER_201_14	The refill detection should be possible only when the parking brake is applied. The parking brake should be steady applied for at least 5 seconds before the vehicles is considered to be parked.
AER_201_15	If a refill is detected the filter algorithm should not be used, the estimate should instead the value indicated by the fuel level sensor(s) until the refill is done (parking brake released). When the refill is ended the algorithm continues to calculate using the current value from fuel level sensor(s) signal as initial value.
AER_201_20	When fuelRate input signal gets incorrect status ("Error or "Not available") the filter algorithm should continue to calculate fuelLevelTot by only using fuelLevel as input. K should be changed to 5*10-5.
AER_201_21	When fuelLevelSensor has status Error or Not Available the output should be the same. The state of Error or Not Available shall not be filtered. When the signal goes from Error or Not Available to a valid sample the filter shall be initialized with the value from the first correct sample.
AER_201_42	If fuelTankSizeLeft and fuelTankSizeRight are both equal to 0 the filter algorithm should continue to calculate fuelLevelTot by only using fuelLevel as input. K should be changed to 5*10-5.

Table 2.4: AER 201 for FLEDS

AER 202	
AER ID	Description
AER_202_2	The lowFuelLevelWarning should be set to 1 (true) when input totalFuelLevel is below a pre-defined level. The level should be 10% for tank sizes equal or below 900liters and 7% for tanks sizes larger than 900liters.
AER_202_3	The lowFuelLevelWarning should be kept true, once it is activated, until the algorithm is restarted by an ECU shutdown or if the totalFuelLevel reaches above 20%.
AER_202_4	The table below describes the requirements for the possible status modes for lowFuelLevelWarning

lowFuelLevelWarning.status	Conditions for status
Not Available	totalFuelLevel.status == N/A || totalFuelLevel.status == Error
Error	-
Good	totalFuelLevel.status == Good

AER_202_5	Output signal lowFuelLevelWarning should have initial value 0 (false).

2.3 HAZARD ANALYSIS TECHNIQUES

The objective of hazard analysis is to identify all the hazards that are caused by malfunctioning behaviour of electrical, electronic, or programmable items. A number of hazard analysis techniques are available nowadays. The techniques that are covered in the next sections are Failure Modes and Effect Analysis, Hazard and Operability Study, and Fault Tree Analysis. The reason why these techniques are explained is because they are in the context of this thesis.

2.3.1 Failure Modes and Effect Analysis

Failure Modes and Effects Analysis (FMEA) is a detailed, bottom up, inductive analysis technique that is used to identify the effects of the primary failure modes of subsystems, components, or functions, and to identify how to control or avoid the undesired effects of these failure modes. FMEA is primarily used for evaluation of the entire system when undesired failure modes occur. However FMEA can be used for hazard analysis as well. The technique provides a possibility to specify failure rates for the primary failure modes, thus serving as a quantitative probabilistic analysis technique. FMEA is oriented towards detailed component and functional level. FMEA is also used to document the analysis and the design changes needed to reduce the risk associated to primary failure modes.

A. **FMEA Process**
Basic FMEA process starts by evaluating the design of the system. After evaluation, possible potential failure modes should be identified along with their possible causes. Effects of the specified failure modes should be considered. If possible, specifying recommended actions to reduce the risk of the potential failure modes is preferable. After following the previous steps, documentation of the results from those steps is essential. The output from FMEA process is a worksheet, as explained in the next paragraph

Figure 2.2: Basic process of FMEA [3]

B. **FMEA Worksheet**
The information retrieved by performing FMEA is usually documented by using a worksheet. Many different types and formats of FMEA worksheets have been suggested by different projects and disciplines over the

past years. The amount and type of information that should be covered in the worksheet is determined by the project particular needs, the person that is performing FMEA, or the safety manager. The minimum information that must be presented in FMEA worksheet that supports hazard analysis is as follows:

- *Failure mode*: This field must be filled in with all the possible failure modes that may affect the item under examination.
- *Failure mode causes*: This field must be filled in with all the possible causes that result in the appearance of a particular failure mode.
- *Immediate effect of failure mode*: This field must be filled in with the direct effect of the failure mode that occurs on the next item in the design.
- *System effect of failure mode*: This field must be filled in with the effect of the failure mode that occurs on the system as a whole.
- *Detection methods of failure modes*: This field describes how the failure mode will be detected. If the method will result in detection of the failure mode before resulting in severe consequences, the detection methods can be specified as mitigation means.
- *Current controls*: This field must be filled in with the controls available currently in the system that are used to prevent failure modes from happening or causing severe consequences.
- *Recommended actions*: This field must be filled in with the methods that can be used for controlling, mitigating, or eliminating the effects of failure modes.

Table 2.5: Basic Sheet of FMEA

Failure Modes and Effect Analysis							
Item	Failure Mode	Failure Cause	Immediate Effect	System Effect	Method of Detection	Current Controls	Recommended actions

C. **Advantages and Disadvantages**

1. Advantages
   - Easy to learn and perform
   - Provides a structural way for evaluating systems, subsystems, and components
   - Allows predicting the reliability of the item under analysis

2. Disadvantages
   - Provide a way to identify failure modes related to multiple components failing together
   - Provides less focus on failures resulting from human errors
   - Requires expertise in order to know what to analyse
   - Boring and time consuming

2.3.2 Hazard and Operability Study

Hazard and Operability study (HAZOP) is a qualitative analysis technique that is used to identify hazards by examining possible deviations in the design of the system [3]. This technique is oriented to subsystems, components, software, environment, and human errors. The analysis can be performed at any level of the design such as conceptual design, high level design, or detailed system design. In order to be able to identify hazards using HAZOP, guide words, parameters, and presentation of the system design are required. Guide words are adjectives that are used to identify deviations in the design. Correct operation of the system is specified by the correct interactions among its components. Interaction among two components is established by passing a parameter, which will affect the correctness of the systems operation. To specify deviations, guide words are combined with parameters. Examples on guide words can be *not supplied when demanded, supplied when not demanded, more, less, early, late, and etc.* Knowledge about the system is necessary in order to be able to identify related deviations and hazards.

A. **HAZOP Process**

HAZOP analysis is conducted by a team and not by a single analyst. Success of HAZOP is determined by the appropriate selection of the team leader and team members. As depicted in Figure 2.3, the basic HAZOP process is constituted of the following steps:

- Establish HAZOP plan-In this step, the analysis goals and schedule are defined.

- Team selection-In this step, the appropriate team leader and team members are selected. Team members should have different disciplines (e.g. design, testing, verification, and etc.)
- Define system elements-In this step, the desired system is divided into smaller subsystems. The decomposition process continues until all the items, component, or functions under analysis have been defines.
- Select guide words-In this step, the required guide words are defined.
- Perform analysis-This step involves many activities such as
  - Identifying the appropriate parameters for every item, component, or function under analysis. The parameters will be the one that will decide the successful operation of the item.
  - Combine each parameter of the item with the appropriate guide words in order to define deviations
  - Derive hazards from deviations
  - Identify consequences for each particular hazard in order to know which hazards have severe consequences
  - If possible, assign corrective actions for the hazards. Corrective actions can be assigned by preventing particular causes from happening
- Document process
  In this step, the entire HAZOP process should be documented by using a worksheet.

Figure 2.3: Basic process of HAZOP [3]

B. **HAZOP Worksheet**
The information retrieved by performing FMEA is usually documented by using a worksheet. The amount and type of information that should be covered in the worksheet is determined by the project particular needs, the team that is performing HAZOP, or the safety manager. Thus

the layout of the worksheet is not critical. The minimum information that must be presented in HAZOP analysis worksheet is as follows:

- *No.*: It's used to identify each analysis line in the worksheet. The column is used for reference purposes.
- *Item*: This field specifies the desired item, component, or function for analysis.
- *Function/purpose*: This field specifies the purpose or the function of the item, component, or function under analysis.
- *Parameter*: This field specifies the parameter of the item, component, or function under the analysis.
- *Guide word*: This field specifies the guide word that will be used with the particular items parameter. It is important to note that more than one guide word can be defined for the particular parameter.
- *Deviation*: This field identifies the particular deviation that results from combining the guide word with the parameter.
- *Consequences*: This field describes the consequences resulting from the identified deviation.
- *Causes*: This field describes the possible and the credible cause for a particular deviation.
- *Hazard*: This field describes the hazard that is resulting from a particular deviation.
- *Risk*: This field specifies the risk for the consequences of the particular hazard. Risk will be measured by combing the severity of the consequences with the probability of occurrence for the particular deviation.
- *Corrective actions*: This field specifies the actions that will be used in order to mitigate or eliminate hazards.

Table 2.6: Basic Sheet of HAZOP

HAZOP analysis										
No.	Item	Function/purpose	Parameter	Guide word	Deviation	Consequences	Causes	Hazard	Risk	Corrective actions

C. Advantages and Disadvantages

1. Advantages
    - The technique is easy to learn and perform
    - Structured and organized technique for hazard analysis
    - Can be applied to any type of system
    - Help the team to discover and think about less obvious behaviours that may result in a deviation

2. Disadvantages
    - Time consuming
    - Considers only single item deviations and not combination of items deviations
    - Depends on the skills of the team

2.3.3 Fault Tree Analysis

Fault Tree Analysis (FTA) is a deductive failure analysis technique that is used to analyse a specific undesired event into its possible causes [3]. It is a top down approach since it starts with a top undesired event down to its causes. Undesired event is a failure that could lead to undesired consequences. Failure means that the system is unable to deliver the required functionality when needed. Failures can cause hazard, where hazard is a source of injury or harm to the environment or people. In reliability and safety engineering, it's important to find the weakness points in the system design when a a specific failure occurs. Therefore, FTA is used to analyse safety-critical failures in those fields. The term undesired event may differ from one field to another. In Aerospace, system failure condition term is used instead of undesired event term. FTA requires a deep analysis of the possible causes for the undesired event. Moreover, the system under analysis might be complex. Therefore, FTA construction process can be exhausting and costly. The top event in the fault tree is the undesired event which is the root of the hazard. Every event that causes the top event should be described in the lower levels of the tree. The lower levels are used to show the relationship between the undesired event and its possible causes. The relationship between the undesired event and its possible cause are modelled using logical symbols and gates. More information about FTA modelling is presented in the upcoming sub sections.

A. **FTA Process**

The elements that are considered necessary in order to construct the fault tree are undesired event and the causes that lead to that event. The top undesired event should be resolved to its intermediate causes. The analysis process of the intermediate causes should continue until the fundamental causes are reached. The top event should provide a description of WHAT the event is and WHEN it will occur. The event can be related to hardware fatigue, component (hardware or software) malfunction, mechanical components malfunction, or a combination of the factors mentioned previously. It's very essential to describe the top event correctly; otherwise it will result in wrong conclusions and evaluation of the systems design.

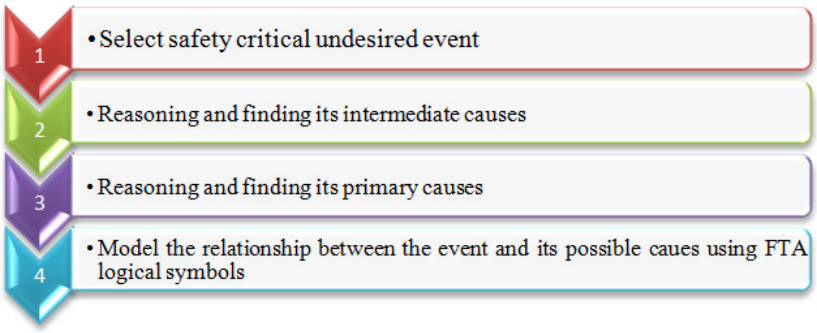


Figure 2.4: Basic process of FTA [3]

B. **FTA modelling**

Modelling symbols are grouped between events, gates, and transfer symbols. Figure 2.5, and Figure 2.6 shows the symbols that are used in this master thesis for modelling FTA. Figure 2.5, shows the event symbols for FTA where there are two types of events that can be modelled, where

- *Basic Event*
  represents the final basic cause for a specific intermediate event.
- *Intermediate event*
  represents the intermediate cause for a specific top undesired event.

Figure 2.6, shows the logical gates that can be used to model the relationship between the top event and its lower level causes. Gates work as follows:

- *AND gate*
  This gate shows that the output event occurs if all input events occur.

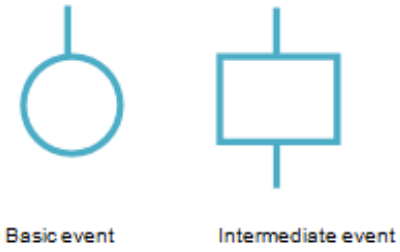


Figure 2.5: Event symbols for FTA

- *OR gate*
  This gate shows that the output event occurs if any of the input events occur.

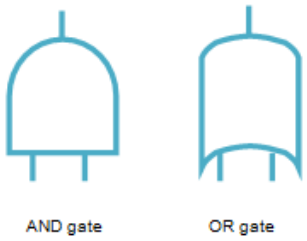


Figure 2.6: Gate symbols for FTA

Figure 2.7, shows an example about a fault tree for the event The car doesn't start during start up. The top event is caused either by a failure in the engine or a failure in the battery. The failure in the battery is caused by an electrical fault whereas the failure in the engine system is caused either by a communication fault or an engine fault. Modelling symbols can vary based on the software tool used for FTA construction.

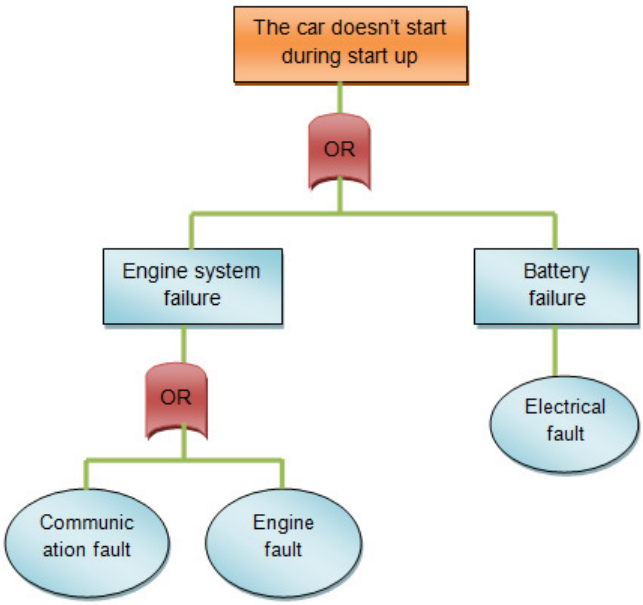


Figure 2.7: FTA example

C. Advantages and Disadvantages

1. Advantages
   - Show the logical relationship between the top event and all its possible causes
   - Evaluate the current design of the system with respect to safety and reliability
   - Find the gaps in the system
   - Suggest effective safety mechanisms that can be used to get rid of the gaps in the system
   - Enhance the testing and maintenance process. Since it provides an understanding about what can cause a failure then this point can enhance the testing process by using fault injection method during testing.

2. Disadvantages
   - Time consuming technique
   - Needs training and experience
   - Could lead to the production of large trees if the domain of the system is broad
   - Modelling dynamic scenarios is hard

- It may not succeed in addressing some problems as it's a binary technique (either fail or success)
- Not possible to analyse combination of events in a single fault tree

Since every technique has its own strengths and weaknesses, it is preferred to use more than one technique for hazard analysis as they complement each other.

2.4 ISO 26262

ISO 26262 is a functional safety standard for road vehicles with a maximum vehicle weight of 3500 kg. The standard specifies a set processes (life-cycles) that have to be followed to achieve safety critical systems. The purpose of ISO 26262 is to reduce the risk of hazards that are caused by a malfunctioning behaviour of electrical, electronic or programmable safety critical systems. ISO 26262 consists of ten parts. Each part in the standard is divided into sub parts that are called clauses. Under each clause there are several requirements. In turn, each requirement has a number of sub requirements. As it can be retrieved from Figure 2.8, ISO 26262 defines a life-cycle that is based on the V-model [ref]. This life-cycle which is followed from left to right is defined in parts 3-7 of the standard. In this thesis, the parts of the standard that mainly are considered are: Concept phase (Part 3), Product development at the system level (Part 4), Product developement at the software level (Part 6). Management of functional safety (Part 2) is also partially considered. Thus, the following subsubsections provide basic information on these parts and clauses.

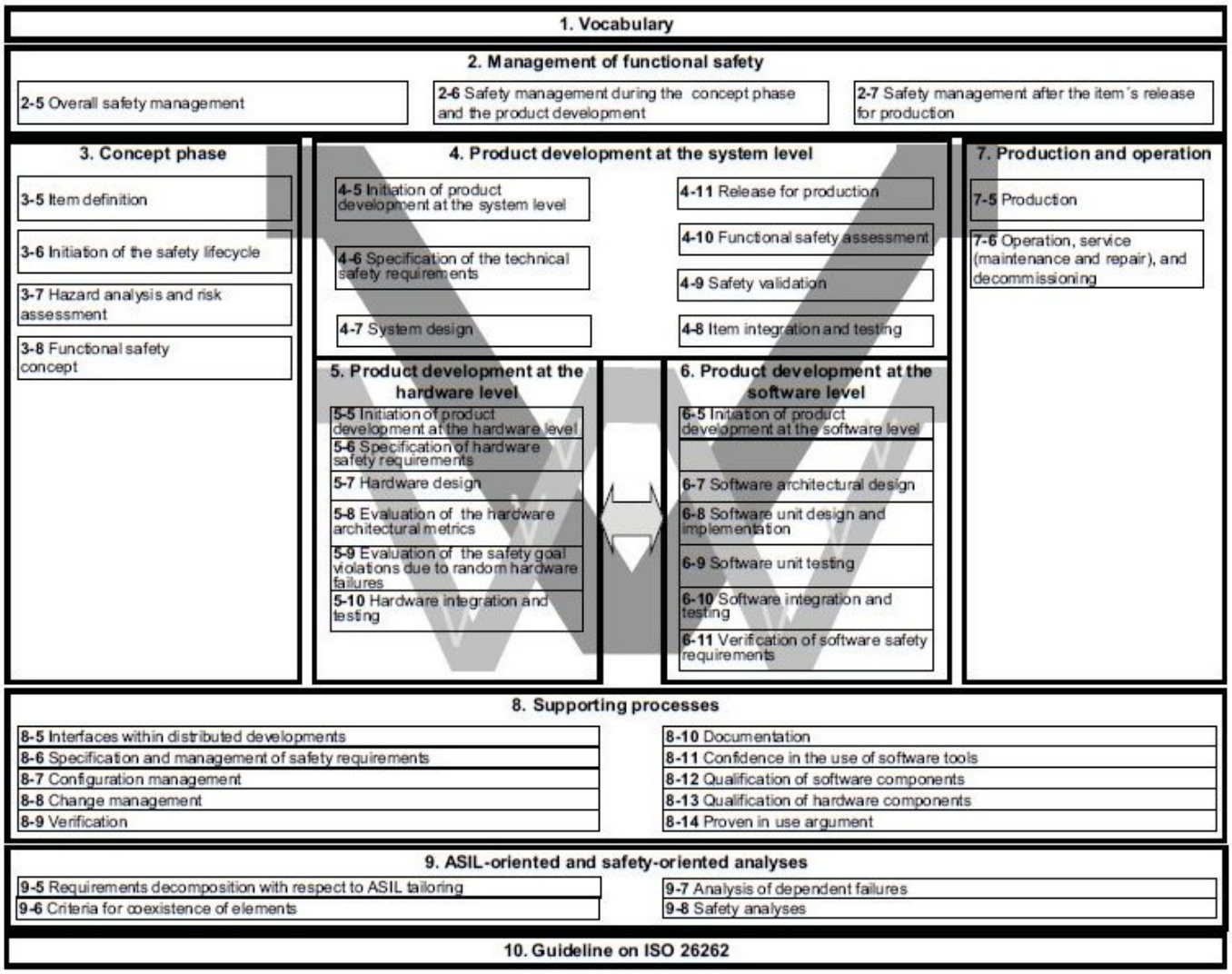


Figure 2.8: Overall structure of ISO 26262 [17]

2.4.1 Management of functional safety

This part is about safety management during different phases of the safety life cycle. It includes three clauses: Overall safety management, safety management during the concept phase and the product development, and safety management after the items release for production. Overall safety management clause specifies the requirements of safety management for the organization that is responsible for the development of the safety related item. One of the requirements prescribes that the organization should have a safety plan along with the process description of the plan. Another requirement prescribes that the organization should have a quality management according to a quality standard as well as a competence management that shows the competence of the people involved in the safety life cycle. The next clause, safety management during the concept phase and product development requires that a safety manager is assigned to be responsible for the whole safety management during Part3 to Part 7. This clause includes other requirements regarding the safety plan that will be followed during these parts. The last clause, safety management after the items release for production specifies the requirements for safety management when the item is released for production in order to ensure that the functional safety is achieved during the production process.

2.4.2 Concept phase

The first clause of this part is **item definition**. Item definition is used to identify the item and all the interactions that the item has with other items and the environment. Item is defined as a systems or a group of systems that implement one or more functionality that is observed at the vehicle level. The main purpose of this clause is to understand what will be developed and how it does work. The next step is to plan how the development according to ISO 26262 is to be made. This is done through **initiation of safety life cycle** clause.
The next step in the concept phase is the **hazard analysis and risk assessment**. In this step, all the hazards that are caused by malfunctioning behavior of electrical, electronic, or programmable items should be identified. Hazard is an undesired event that results in harm to the humans or the environment. For each identified hazard, an automotive safety integrity level (ASIL) should is assigned. ASIL taxonomy consists of four levels that range from A to D, in which D is the highest and A is the lowest. ASIL is used to know which requirement of ISO 26262 to follow in order to reduce or avoid the risk of particular hazards. For each particular hazard that has an ASIL level A, B, C, or D, at least one safety goal is identified. A safety goal is a sentence that describes how to avoid or mitigate a particular hazard. The safety goal inherits the ASIL level of the particular hazard that it aims

to mitigate or avoid. The following is the requirements of hazard analysis and risk assessment that has been covered in this thesis:

- 7.4.2.2.1 *The hazards shall be identified systematically by using a suitable analysis technique.*
- 7.4.2.2.2 *Hazards shall be specifed in a way that can be observed at the vehicle level.*
- 7.4.2.2.3 *The hazardous event shall be specified by comining pertinent hazards and operational situations.*
- 7.4.2.2.4 *The consequences for each hazard shall be specified.*
- 7.4.3.1 *All the specified hazardous events shall be classified.*
- 7.4.3.2 *For each hazardous event, the severity level shall be specified to one of the four levels S0, S1, S2 or S3 with respect to Table 2.7.*
- 7.4.3.4 *TFor each hazardous event, the exposure level shall be specified to one of the four levels E0, E1, E2, E3 and E4, with respect to Table 2.8.*
- 7.4.3.7 *For each hazardous event, the controllability level shall be specified to one of the four levels C0, C1, C2 and C3 with respect to Table 2.9.*
- 7.4.4.1 *For each identified hazardous event, an ASIL level is assigned using the parameters "severity", "exposure" and "controllability" with respect to Table 2.10.*
- 7.4.4.3 *A safety goal shall be specified for each hazardous event with an ASIL level A to D. If similar safety goals are specidied, these may be combined into one safety goal.*
- 7.4.4.4 *The safety goal shall inherent the same ASIL level for the corresponding hazardous event. If similar safety goals are combined into a single one, with respect to 7.4.4.3, the combined safety goal shall inherent the highest ASIL level.*

The next step in the concept phase is to define at least one functional safety requirement for each safety goal. Functional safety requirement is a sentence that describes the functionality to achieve the safety goal but it shouldnt describe how it will be implemented in hardware or software. Each functional safety requirement inherits the ASIL level of the particular safety goal. Allocation of functional safety requirement to the preliminary architectural elements of the item is essential. The derivation of functional

Table 2.7: Classes of severity

	Class			
	S0	**S1**	**S2**	**S3**
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries

Table 2.8: Classes of exposure

	Class				
	E0	**E1**	**E2**	**E3**	**E4**
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Table 2.9: Classes of controllability

	Class			
	C0	**C1**	**C2**	**C3**
Description	Controllable in general	Simply controllable	Normally controllable	Difficult to control or uncontrollable

Table 2.10: ASIL classification

Severity class	**Probability class**	**Controllability class**		
		C1	**C2**	**C3**
S1	**E1**	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	**E1**	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	**E1**	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

safety requirements, ASIL, and their allocation to the preliminary architecture is called **functional safety concept**. The requirements that has been covered in this thesis are as follows:

- 8.4.2.1 *The functional safety requirements shall be deduced from the safety goals, taking into account the elementary architectural presumptions.*
- 8.4.2.2 *At least one functional safety requirement shall be identified for each safety goal.*
- 8.4.3.1 *Every functional safety requirement shall be assigned to the elementary architectural presumptions.*

2.4.3 Product development at the system level

The first clause in this part is planning of the work activities and it's called **initiation of product development at the system level**. The next clause is to the **specification of technical safety requirements** for each functional safety requirement. Technical safety requirement specifies how to implement a functional safety requirement in hardware or software. Allocation of technical safety requirement to hardware and or software is essential. Next step is to develop the **system design** based on the functional safety requirements, technical safety requirements, and non safety related requirements. After developing the system design, verification of the system design shall be conducted with respect to the Table 2.11. Afterwards, the software architectural design shall be developed. The software architectural design shall be verified by using the verification methods listed in Table 2.12 [17]. Afterwards **product development at the hardware respective software** is conducted in which these parts are explained later in this thesis.

Table 2.11: System design verification methods [17]

Methods		ASIL			
		A	B	C	D
1a	System design inspection[a]	+	++	++	++
1b	System design walkthrough[a]	++	+	o	o
2a	Simulation[b]	+	+	++	++
2b	System prototyping and vehicle tests[b]	+	+	++	++
3	System design analyses[c]	see Table 1			

[a] Methods 1a and 1b serve as a check of complete and correct implementation of the technical safety requirements.

[b] Methods 2a and 2b can be used advantageously as a fault injection technique.

[c] For conducting safety analyses, see ISO 26262-9:2011, Clause 8.

Table 2.12: Verification methods for the software architecture design [17]

	Methods	ASIL			
		A	B	C	D
1a	Walk-through of the design[a]	++	+	o	o
1b	Inspection of the design[a]	+	++	++	++
1c	Simulation of dynamic parts of the design[b]	+	+	+	++
1d	Prototype generation	o	o	+	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis[c]	+	+	++	++
1g	Data flow analysis[c]	+	+	++	++

[a] In the case of model-based development these methods can be applied to the model.

[b] Method 1c requires the usage of executable models for the dynamic parts of the software architecture.

[c] Control and data flow analysis may be limited to safety-related components and their interfaces.

The next step after product development at the hardware and software level is **item integration and testing**. The purpose of item integration and testing is to test every safety requirement in order to see if it is in compliance with its specification and ASIL categorization. Moreover, item integration and testing is used to verify that the items within the vehicle interact correctly. Item integration and testing is conducted at three levels: *hardware-software, item, and vehicle level.* The first level is to integrate and test the hardware and software of each particular element that compose the item. After that, integration and testing of all the elements that compose a particular item is conducted. Thereafter, integration and testing of all the items that compose the vehicle is conducted.

Thereafter in this part is **safety validation**. The purpose of safety validation is to provide evidence based on tests that the safety goals have been realized at the vehicle level. Moreover, safety validation is used to check whether the functional safety of the item is in compliance with the functional safety concept.

After that, **functional safety assessment** is conducted in order to evaluate the functional safety accomplished by the item. Thereafter, **release for production** clause is conducted. In this step, a check is made to see if the item meets all the considered requirements for functional safety. In case the item meets its requirements, release for production can start.

2.4.4 Product development at the software level

The first step starts with the planning of the work to be performed during this part. This step is called **initiation of the product development at**

the software level. The next step is to derive **software safety requirements** from the technical safety requirements along with the allocation of these requirements to software parts in a high abstraction level. Thereafter, **software architectural design** that meets software safety requirements is developed. The next step is to **design and implement** software units that are in compliance with the architectural design and the software safety requirements. Verification of the designed and implemented software units is followed. Next step is the **software integration and testing** in which is used to integrate all the software units and test them to demonstrate that the designed software architecture meets its software safety requirements. The last step is the **verification of software safety requirements**. The purpose of the verification of software safety requirements according to ISO 26262 is to demonstrate that the embedded software satisfies its requirements in the target environment [17].

2.5 SAFETY CASE

A safety case is defined as an argument supported by evidence to show that the system is safe enough to operate in a given context [14]. The safety case should be understandable by different stakeholders, convincing about the safety of the system, as well as complete and consistent. The main elements of a safety case are requirements, evidence, argument, and context. More explanation about each of the elements is provided as follows:

- Requirements: are the safety requirements, goals, or objectives that must be achieved in order to ensure the safety of the system.
- Evidence: The evidence about the safety of the system. The evidence can be based on the development process, testing, verification, simulation, safety management, and analysis.
- Argument: It's used to connect safety requirements to their evidence in a structured and a manageable way.
- Context: Identifies the domain or scope within which the safety to be argued.

It is mandatory for the safety case to have these four elements otherwise the case is not complete. An argument without evidence is baseless whereas evidence without an argument is unexplained [33]. The elements of the safety case are inter-related. Thus consistency must be maintained when a change is introduced in any of the safety case elements. For example, if a change has been introduced to the context, other elements should be checked to maintain if they are still valid in the context. Figure 2.9 shows

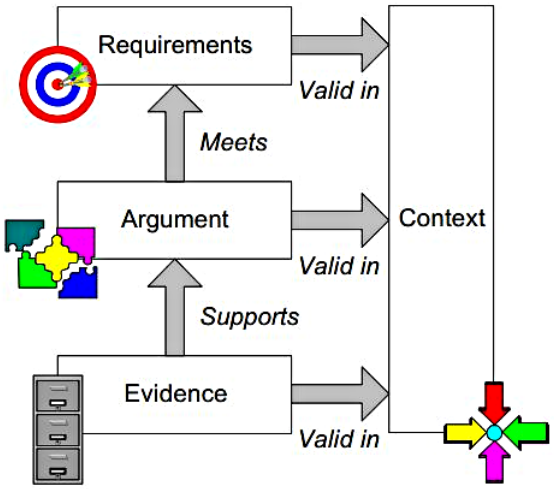


Figure 2.9: Dependencies among the safety case elements [33]

the elements of the safety case as well as the dependencies among these elements. The figure shows that the elements should be valid in the context.

Based on the type of the evidence, the argument will be classified as either process-based or product-based. Process-based argument assumes that a good process will lead to a good product. Process-based safety case should provide evidence about the ordinary development process as well as the safety engineering process followed. Thus process-based argument can have two sub-arguments. The first sub-argument should provide a set of evidence to show that the development process used in the development life cycle is rigorous. The second sub-argument should provide a set of evidence to show that the safety engineering process used is effective. The other type of safety cases is the product-based argument. product-based argument should provide evidence that is related to the safe behavior of the product. Product-based evidence should show that the system has the required safe behavior when something wrong happens. The system should have fail-safe techniques. Fail-safe techniques result in a product that is more capable in handling all the considered failures. Fail-safe techniques can be either hazard mitigation or elimination techniques. These techniques should be considered in the design phase of the product. product-based argument should be supported by evidence from testing, verification, and simulation if possible.

2.5.1 Safety Case in Compliance with ISO 26262

Before the introduction of ISO 26262, the automotive industry has followed IEC 61508 international standard for electrical and electronic systems [16] and MISRA Guidelines for Safety Analysis of Vehicle Based Programmable Systems for development, operation and maintenance of safety electrical and electronic embedded systems. With the introduction of ISO 26262, requirements about the safety case have been stated clearly in the standard. A safety case in compliance with ISO 26262 (as stated in Part 2, Management of Functional Safety [17]) should meet the following requirements:

"**6.4.6.1** *The safety case shall be complied with for items that have at least one safety goal with an ASIL (A), B, C or D: a safety case shall be developed in accordance with the safety plan*".

"**6.4.6.2** *The safety case should compile the work products that are generated during the safety life cycle*".

The requirements means that the safety case should be developed for items with an ASIL level A or higher and it shall consist of a set of work products that are generated by following the activities of ISO 26262 where work products means the result of performing one or more requirements of the standard . Requirement 6.4.6.2 encourages box ticking mentality as the company or organization may pretend that it has a safety case in compliance with the standard just because it has the required work products.

2.5.2 Safety Case Life Cycle

A safety case is usually built at the end of the development life cycle. However this approach has a number of disadvantages such as:

- The resulted safety case is less robust as the safety case developers will have to argue over the design as it's given to them. Thus it will not be possible to influence the design in order to improve safety.
- The resulted safety case could lead to redesign and redevelopment of the system, in which it can be expensive with respect to time and money.

It is required by some safety standards such as U.K. Defense Standards 00-56 [26] to develop safety case during different phases of the development life cycle. Three versions of safety cases can be obtained based on in which phase the case is presented. These versions are as follows [33]:

- Preliminary Safety Case At this stage, the safety case is presented after the systems requirements have been identified and reviewed. The

safety case in this stage will present the objectives, the arguing approach as well as the anticipated evidence.

- Interim Safety Case At the interim stage, the safety case will be updated to reveal the knowledge from detailed design and system specification.
- Operational Safety Case At the operational stage, the safety case will be updated with complete evidence that shows that the system meets its safety requirements. This safety case should be created before the system is put into service. Figure 2.10 shows the safety case development during traditional development life cycle.

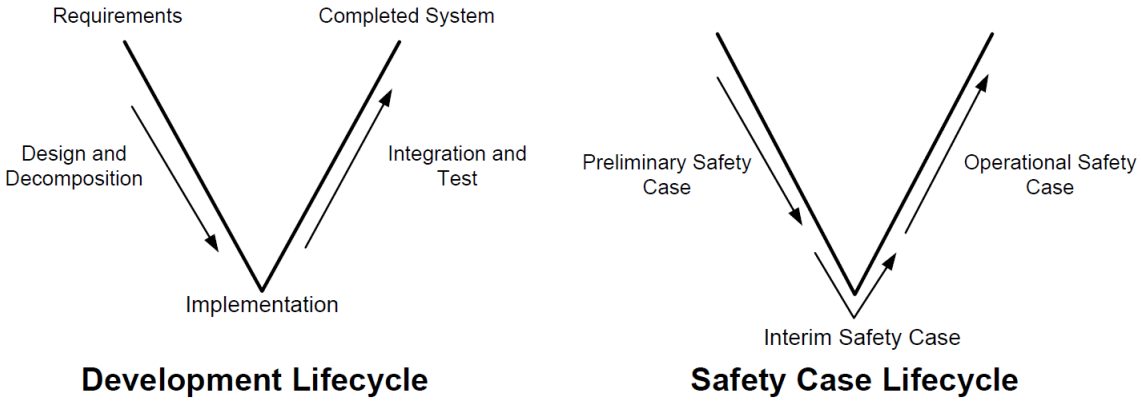


Figure 2.10: Safety case life cycle during traditional development life cycle [33]

2.6 Modelling Techniques

Based on the presentation of information used for describing the safety case arguments, modeling techniques can differ between graphics-based notations and text-based notations.

2.6.1 Text-Based Notations

This type of notation uses textual presentation to represent the elements of the safety case. Normal Prose, Structured Prose, and Argument Outline are some styles used for text-based notations. In normal prose, the safety argument is written as a normal text as presented in Figure 2.11. It can be difficult to understand and trace the structure of the argument in normal prose. One way to solve this problem is to use structured prose, in which it requires to denote the structure of the argument by highlighting claim, context, evidence, strategy, solution, assumption, and justification. Figure 2.12, shows an example of a structured prose. The structure of the argument can be made clearer by using argument outline style. Different formats can be used in argument outline. Figure 2.13, shows an example of argument outline style. Other text-based notation styles are available such as mathematical proof and lisp style. Refer to [4] for more information about these styles.

The control system is acceptably safe, given a defintion of acceptably safe, because all the identified hazards have been eliminated or sufficiently mitigated and the software has been developed to the integrity level appropriate to the hazards involved.

Given both the tolerability targets of the hazards (from reference Z) and the list of hazards identified from the functional hazard analysis (from reference Y), we can show that all identified hazards have been identified or sufficiently mitigated by arguing over all the three of the identified hazards: H1, H2, and H3.

.....

Figure 2.11: Normal prose example

2.6.2 Graphics-Based Notations

This type of notation uses graphical presentation to represent the elements of the safety case. Claim Argument Evidence (CAE), and Graphical Structure Notation (GSN) [36] are the notations used for graphical presentation of the safety case arguments. CAE has a number of modeling elements that represent claim, argument, evidence, and the relationships among these elements. The basic argumentation elements that are used in modeling safety case arguments using CAE are given in Figure 2.14.

The argument establishes the following **claim**: the control system is acceptably safe, within the **context** of acceptably safe. To establish the top-level claim, two **sub-claims** are established: (1) all the identified hazards have been eliminated or sufficiently mitigated and (2) the software has been developed to the integrity levels appropriate to the hazards involved.

Within the **context** of the tolerability targets of the hazards (from reference Z) and the list of hazards identified from the functional hazard analysis (from reference Y), we follow the **strategy** of arguing over all the three of the identified hazards: (H1, H2 and H3).
.....

Figure 2.12: Structured prose example

Claim 1: Control system is acceptably safe.
Context 1**:** of acceptably safe.

Claim 1.1: All the identified hazards have been eliminated or sufficiently mitigated.
Context 1.1-a: Tolerability targets for hazards (reference Z).
Context 1.1-b: Hazards identified from the functional hazard analysis (from reference Y)

Strategy 1.1: Argument over all identified hazards (H1, H2, H3)
.....

Figure 2.13: Argument outline example

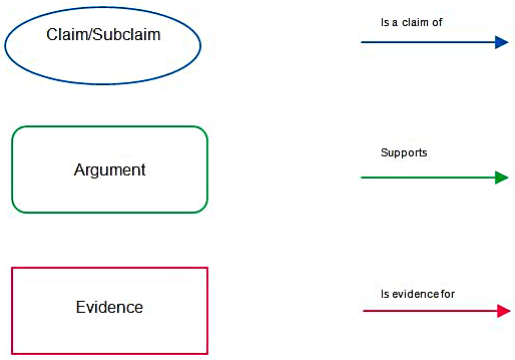


Figure 2.14: CAE modelling elements

GSN uses basic argumentation elements to model the individual elements of the safety case. GSN modeling elements are explained below, and presented in Figure 2.15.

- Goal: is a claim about the system
- Strategy: A method that is used when decomposing a claim or goal into sub claims or sub goals.
- Solution: represents evidence that shows that a particular goal or claim has been met.
- Context: definition of the domain or scope in which a goal, evidence or strategy is given
- Undeveloped entity: Indicate that specific part of the argument has not been developed. It can be applied to goals and strategies.
  - Undeveloped goal: It indicates that the goal has not been developed yet because the evidence supporting this goal is not available yet.

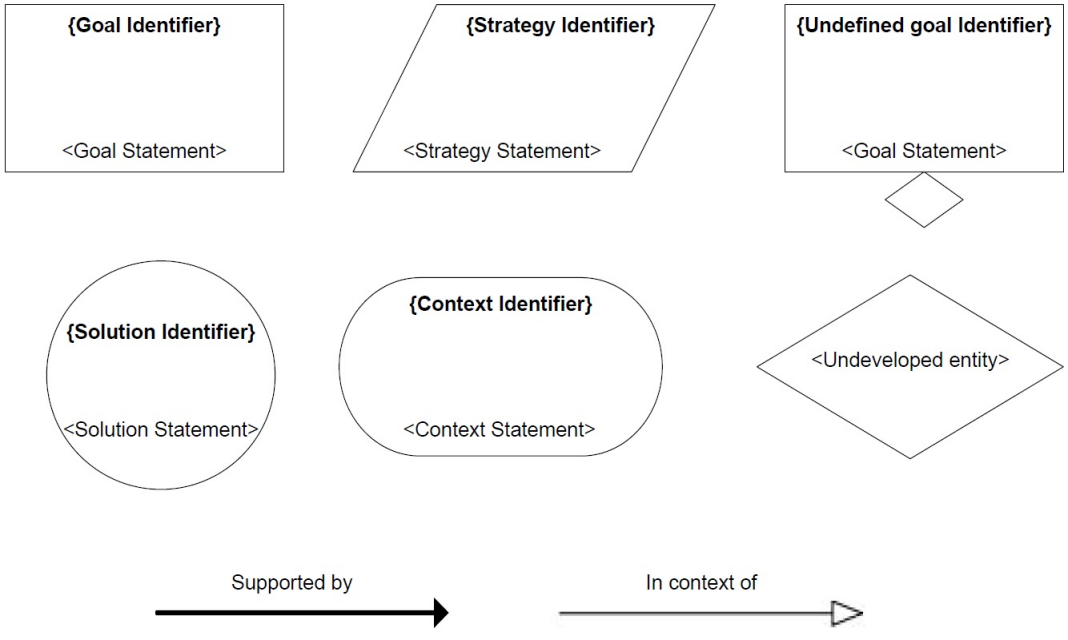


Figure 2.15: GSN modelling elements [9]

The argument is built by linking together the basic elements using two relationships that are *solved by* and *in context of*. The claim is continuously

decomposed into sub claims using GSN strategy element. The decomposition continues until sub claims are supported by direct evidence. Linking together GSN elements in a network is called a goal structure. Figure 2.16, shows an example about goal structure. In this example, the top level goal G1 "The control system is acceptably safe" in the context C1 "Definition of acceptably safe" is broken down into sub goals G2 "The possible hazards have been identified" and G3 "The possible hazards have been mitigated" by using strategy S1 "Argument over hazard identification and mitigation". G2 and G3 are supported by direct evidence sol1 "Hazop" and sol2 "Verification" respectively.

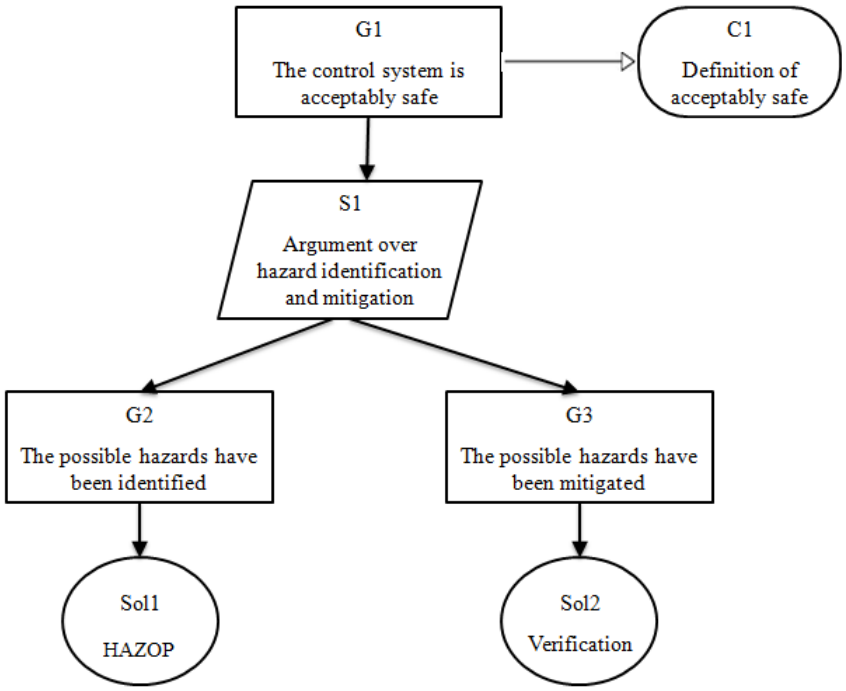


Figure 2.16: Goal structure example [35]

GSN has two extensions: Patterns and Modular extensions [9]. Patterns are used for recording and documenting successful argument structures for reusability purposes [9]. The pattern can be reused when building a safety case for a system that has a similar domain as the pattern's domain. The pattern should have a description that demonstrates the purpose, motivation and other information necessary about the pattern. It could result in improper use of the pattern in case those necessary information are missing. Figure 2.17 shows the extensions of GSN

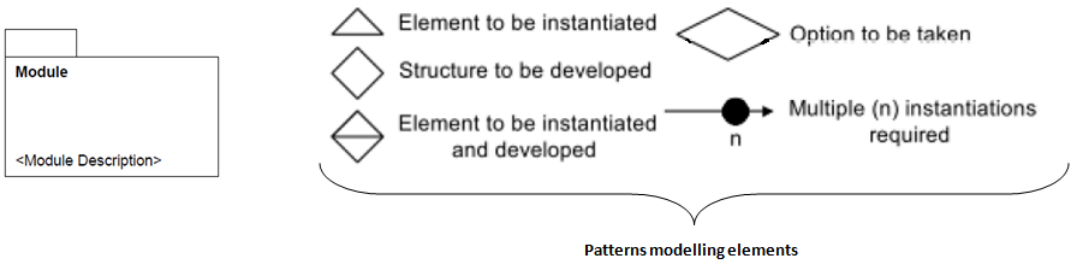


Figure 2.17: GSN extensions [9]

Figure 2.18, shows a functional breakdown pattern as an example on GSN patterns. This pattern argues the safety of the system functions by arguing further that each of those functions are safe and by arguing that either the interaction among the system function are independent or the system functions are independent from each other (no interactions). This pattern is used in chapter 6 to construct the safety case of FLEDS. A safety case can grow in size and complexity for complex safety critical systems. Thus modularization can be used to construct modules of safety arguments.

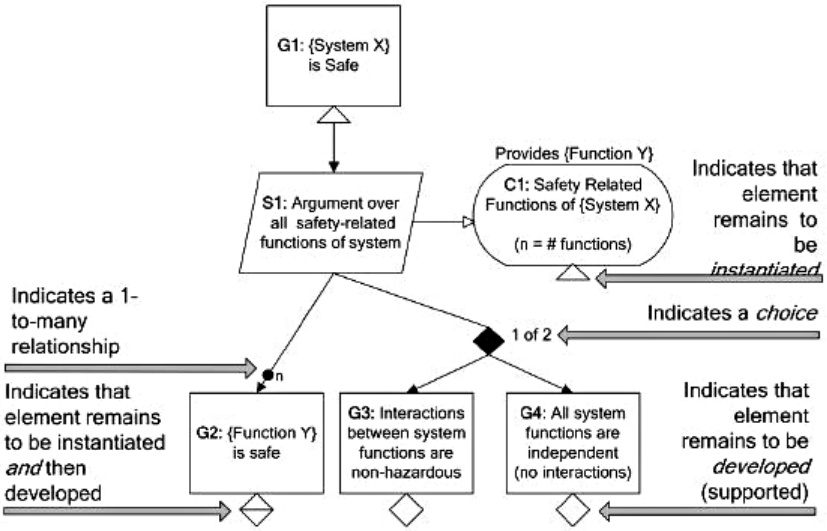


Figure 2.18: Functional breakdown pattern [35]

2.7 Safety Case Fallacies

Flaws in the safety case are called fallacies. Undetected fallacies could lead to over trust of the systems safety. Therefore, fallacious argument could result in a system that has modes that could result in accidents. Fallacies are widespread in safety arguments and in order to be able to detect them, awareness about them is required. Thus, logical fallacies have been categorized into the following [37]:

- Circular reasoning: Happens when the argument reassert its claim in a way that it makes it true.
- Diversionary arguments: contain enormous amount of insignificant material in order to divert the readers attention from a feeble supported claim.
- Unsupported assertions: occur when an argument is not supported by evidence.
- Anecdotal arguments: concern arguments in which their claims are true in particular circumstances.
- Fallacious appeals: concern arguments that are supported by irrelevant evidence.
- Mathematical fallacies: contain common defects in mathematical inferences.

- Omission of key evidence: occurs when the argument lacks the necessary evidence to ascertain its authenticity
- Linguistic fallacies: Concerns arguments where the language is weak and distracting where such language could direct the reader to undesired conclusions.

A safety case expert can compare his arguments with respect to the different types of fallacies in order to see if the arguments have any underlying logical fallacies.

2.8 Challenges when Developing Safety Cases

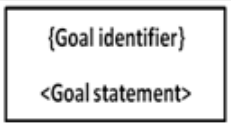
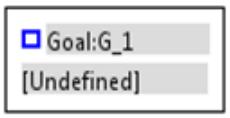
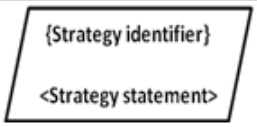
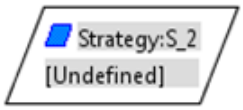
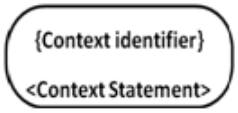
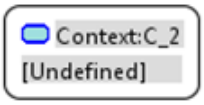
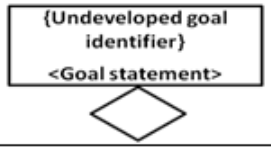
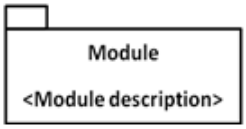
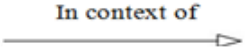
Many challenges could appear while developing safety cases. Such challenges are:

- The person that is responsible for building the safety case can tend to show information that support safety and hide information that shows the opposite. This is called confirmation bias
- There's a risk that the safety case becomes a paper practice in case the regulator demonstrate the required work products that the company has without actually checking the safety of the system.
- Safety case can increase in size and complexity for complex systems. Thus making it difficult to understand
- It can be difficult to organize and introduce evidence from different sources. Thus it's difficult to keep up a clear structure of the safety argument
- It can be difficult for the reader to find where the relevant safety information can be found in the safety document

2.9 D-Case editor

D-Case editor has been used for the creation of the safety case. The goal of this thesis was not to make a survey between different tools due to time limitation of this thesis. D-Case is an editor for dependability cases where D stands for dependability. It's implemented as an eclipse plugin and it has a framework for benchmark test [10]. The author of this thesis has decided to choose D-case editor because it is a user friendly and it provides support for modularity and GSN. Table 2.13 shows the mapping of GSN elements to D-Case elements.

Table 2.13: Mapping of GSN elements to D-Case elements

GSN modelling elements	D-Case modelling elements
	
	
	
	
	
	
	
	
	

2.10 Related Work

Based on our experience, there are few studies regarding the experience of building a safety case and certifying systems in compliance with ISO26262. Born et al. in [20] demonstrate the experiences from applying ISO26262 to a German car manufacturer. Even though the paper is not considering the issue of building a safety case, however, the experiences are important with respect to building a safety case in compliance with ISO26262. Born et al. experiences present three issues of applying the standard to car manufacturers. The first issue is that the company is accustomed to the existing internal safety processes and unwilling to the external obliged processes. The second issue is that the company is concentrating on the documentation rather than the contents of the documents. Moreover, the companies having difficulties in maintaining consistency among multiple documents and versions. This issue leads to the third issue; the problem of maintaining traceability among multiple documents. Born et al. suggests that traceability can be maintained by using cross-referenced identifiers among requirements, hazards and etc. The first issue of Born et al. is in consistent with the results of Kienle et al. presented in [1]. Kienle et al. results were based on an industrial questionnaire. The questionnaire showed that the internal code guidelines and processes are more important than the external ones. Johansson et al. in [18] demonstrate the gap analysis between the development life cycle of Scania and the requirements of ISO 26262. The gap analysis was conducted in order to define what is needed by Scania to achieve compliance with ISO 26262. The analysis of Johansson et al. showed that: No ASIL classification is conducted by Scania, requirement derivation by Scania is not performed as required by ISO 26262, and no planning for safety activities is conducted by Scania as required by ISO 26262. Johansson et al. suggests that Scania must focus on ASIL classification and the derivation of requirements in case Scania is interested in compliance with ISO 26262. ASIL classification and the derivation of requirements are essential sections of the standard. Following other sections of ISO 26262 are dependent on both of those sections. Feather and Markosian in [21] present their experiences about building a safety case for a piece of a safety-critical software that is used to launch a Nasa's vehicle. Their experiences present two issues. The first issue is that it was very difficult for them to start constructing the safety case as there were no examples about safety cases for software systems. The second issue is that it was hard to achieve a well-written and well-structured safety case. The authors of the safety case didn't have a good understanding of how safety cases should be structured. In [6], Törner and Öhman present an industrial study regarding the introduction of the safety case in the automotive industry. Their study was based on interviews and workshops. The first issue from their study showed that the main concept of the safety case was not used. Moreover, the results showed that it was difficult to collect

information for building safety cases. Moreover, competence and experience are required when building safety cases. In this thesis, similar issues were experienced as presented in chapter 7.

3. PROBLEM FORMULATION

This thesis consider applying ISO 26262 and building a safety case on an industrial setting. The author of this thesis will try to show how the different work products that are generated during the development life cycle can be used to construct the safety case. Moreover, throughout this thesis we will try to address the following questions:

- What does it mean to build a safety case in compliance with ISO 26262? This question direts us to the following subquestions:
  - How to make the safety case in compliance with the standard?
  - How to find the evidence relevant to build the safety case?
- What does it mean to have a clear separation between process and product-based arguments? This question opens the door to another one which is; what makes a work product a process-based evidence and what makes it a product-based evidence?
- What are the main challenges that may appear when applying ISO 26262 and building the safety case?

4. SOLUTION METHODS

This chapter presents the solution methods that have been used to perform this thesis as well as the scope of this thesis. Moreover, the limitations of this thesis with respect to the time limit and the documents that has been found for the system are presented further in this chapter.

In order to be able to perform this thesis in a structured way, a number of steps have been taken. The first step is to study ISO 26262 in order to be able later to map what has been found for the system to the standard in order to produce process-based evidence that is in compliance with the standard. ISO 26262 study has been conducted in section 2.4. Thereafter, a detailed study of the system under analysis is conducted as presented in section 2.2. If information needed about the system is missing, interviews are conducted to gather relevant information (information gathering). Thereafter, evidence about the safe behaviour of the system and the process that has been followed during the system development are collected as presented in chapter 5. The next step is to map what the system has to the standard and in case of absence of important evidence, evidence should be provided. The mapping is presented in chapter 5. The last step is to search and find a tool for constructing the safety case as presented in section 2.9. Figure 4.1 shows these steps. The resulting safety argument can be ambiguous and un-understandable because not all engineers are able to write clear, well reasoned, and well structured English. To overcome the limitations of text-based notations, graphical structure notations are used as explained before in subsection 2.6.2. GSN is the one that has been used in this thesis because of:

- Ease of construction and management of the safety argument
- Easy to understand the logical flow in the safety argument
- Ease of maintenance
- Tool support
- Modularity and patterns support

To make the construction of the safety case easier, a tool that supports modularity is needed. Thus, searching and finding a suitable tool is essential.

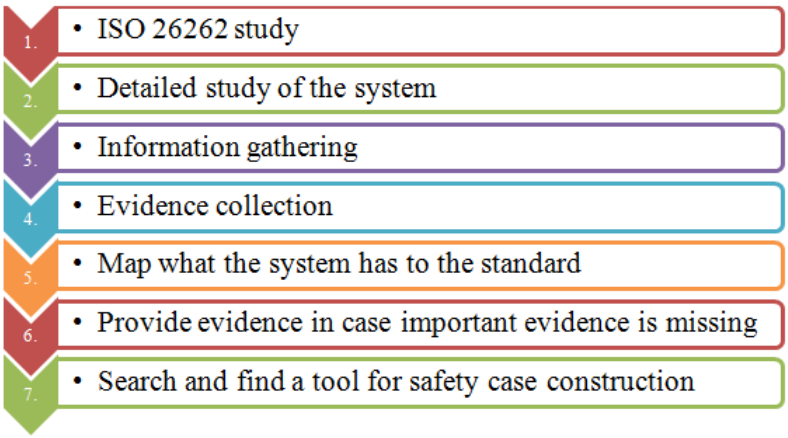


Figure 4.1: Solution methods used in this thesis

4.1 Scope

The scope of this project is to develop an operational safety case for FLEDS because the system is already in the production phase and the evidence regarding the system safety is present.

4.2 Limitations

- With respect to the time limit of the thesis and with respect to what documents have been found regarding the system, the work covers only the software functions of the system. Therefore, the hardware of the system haven't been analysed with respect to safety.
- The parts of ISO 26262 that has been covered in this thesis, are based on what have been found for the system under analysis. The system is studied and each activity that is implemented for the system is checked against ISO to see to which part of the standard it corresponds.

5. COLLECTION AND PROVISION OF EVIDENCE

To build a safety case, first of all, the claims to be supported must be clear and the evidence to support them must be identified. Usually, the top-level claim is that the system is acceptably safe with respect to the definition of acceptably safe. This top-level claim is shown to be founded by providing evidence that all the hazards that lead to intolerable risk are mitigated. This top-level claim stems from the objective of ISO 26262, which states that the product should ensure an adequate and acceptable level of safety. Our work has not consisted in making a cost and benefit analysis to achieve a definition of acceptably safe for which the risk is as low as reasonably practical. Instead, we have proceeded as if a definition was present or at least could be provided in further developments of this work. Our main focus has been in finding a clear mapping between the evidence required by the standard and the evidence available in the company. This mapping was needed to understand which evidence could be collected to build the safety case and which evidence was missing and needed to be provided. To do such mapping, we thoroughly studied FLEDS, ISO 26262 and the safety life-cycle adopted by the company. Moreover, interviews have been conducted with the employees who were involved in the development of the system. Therefore, in this section, a mapping of what the system has to the standard is presented. The clauses that are covered in this section are based on what have been found and or provided for the system regarding hazard analysis, design, testing, verification and etc.

5.1 Hazard Analysis and Risk Assessment (Pt 3, Cl 7)

As stated in the background subsection 2.4.2, the objective of hazard analysis and risk assessment is to identify and classify all the hazards that are caused by malfunctioning behaviour of electrical, electronic, or programmable items. No hazard analysis and no ASIL classification have been made in Scania since the standard is not adopted. Thus, evidence about hazard analysis, ASIL classification and safety goal identification was essential. Thus this evidence has been provided by the author of this thesis. HAZOP technique has been used for hazard analysis and more fields to the HAZOP sheet have been added in order to make it in compliance with ISO 26262. UFR of FLEDS corresponds to SG. Thus a mapping from UFRs to SGs is conducted for the specification of SGs. Both, the requirements of hazard analysis and risk assessment in the background subsection 2.4.2 and the hazard analysis techniques suggested in the course DVA321 [5] has been followed when providing an adapted HAZOP table.

Hazard analysis has been conducted for the main two functions of FLEDS (fuel level estimation and low fuel level warning). Adapted HAZOP technique has been selected for hazard analysis. Table 5.1, shows HAZOP analysis for FLEDS. To make ASIL classification much more clear, colours has been used such as red, orange and green where red stands for ASIL level D (catastrophic), orange stands for ASIL A, B, and C, and green stands for No ASIL level or QM level. Safety goals have been formulated as it is required in requirement 7.4.4.3 and 7.4.4.4 in the background subsection 2.4.2. Table 5.2, shows the safety goals that are formulated for each hazard.

Table 5.1: Hazard Analysis using Adapted HAZOP [5] for FLEDS

Adapted HAZOP												
ID	Function	Parameter	Guide-word	Deviation	Hazardous event	Operational situation	Conseq	S	E	C	ASIL	Causes
H1	Fuel level estimation (AE201)	Total fuel level	Supplied too high	Total fuel level supplied too high	Fuel gauge Indicates higher fuel level than actual fuel level in the tank during driving	Free way	Vehicle is driven until no more fuel could be collected from the tank. Resulting in engine stop suddenly. Thus crush by other cars coming from behind is expected.	3	2	2	A	1) Erroneous fuel estimation by Kalman filter 2) Bug in gauge function 3) Mechanical Fault in gauge
						High way- heavy traffic		2	3	1	QM	
						City driving, slippery road-high traffic		2	3	2	A	
						City driving-snow and ice-driving speed 50 km/h		3	2	3	B	
H2	Fuel level estimation (AE201)	Total fuel level	Supplied too low	Total fuel level supplied too low	Fuel level gauge Indicates lower fuel level than actual fuel level in the tank during driving	Free way	Annoyed driver-trust issues for future fuel level display.	0	2	0	NO ASIL	1) Erroneous fuel estimation by Kalman filter 2)Bug in gauge function 3)Mechanical fault in gauge
						High way- free with wet roads		3	4	3	D	
						City driving, slippery road-high traffic		2	3	2	A	
						City driving-snow and ice-driving at 50 km/h		3	2	3	B	
H3	Fuel level estimation (AE201)	Total fuel level	Not supplied when demanded	Total fuel level not supplied when demanded	Fuel gauge indicates no fuel level when it should not during driving	Free way	Vehicle is driven until no more fuel could be collected from the tank. Resulting in engine stop suddenly. Thus crush by other cars coming from behind.	3	2	2	A	1)Mechanical fault in gauge 2)Hardware fault in ICL ECU system 3)Bug in gauge function 4)No total fuel level received by ICL ECU system
						High way-free driving with wet roads		3	4	3	D	
						City driving, slippery road-high traffic		2	3	2	A	
						City driving-snow and ice-driving speed 50 km/h		3	2	3	B	
H4	Low fuel level warning (AE202)	Low fuel level indicator	Supplied when not demanded	Low fuel level indicator supplied when not demanded	Fuel level warning displayed when it shouldn't.	Free way	Annoyed driver-trust issues for future fuel level warnings.	0	2	0	No ASIL classification required	1) Erroneous output from low fuel level warning function 2) Bug in warning lamp function in ICL
						High way-free driving with wet roads		0	4	0		
						City driving, slippery road-high traffic		0	3	0		
						City driving-snow and ice-driving 50 km/h		0	2	0		
H5	Low fuel level warning (AE202)	Low fuel level indicator	Not supplied when demanded	Low fuel level indicator not supplied when demanded	Fuel level warning not displayed when it should	Free way	Vehicle is driven until no more fuel could be collected from the tank. Resulting in engine stop suddenly.	3	2	2	A	1)Erroneous output from low fuel level warning function 2) Bug in warning lamp function in ICL 3)Communication problem between COO and ICL 4)Fault in warning lamp
						High way-free driving with wet roads		3	4	3	D	
						City driving, slippery road-high traffic		2	3	3	A	
						City driving-snow and ice-driving speed 50 Km/h		3	2	3	B	

Table 5.2: Safety goals for FLEDS

Safety Goals				
Hazard ID	ASIL	Safety Goal Description	Safety Goal ID	Corresponding UFR
H1, H2	D	The indicated fuel level shall not deviate more than or less than 5% from the actual volume in the tank	SG1	UFR18_1
H3	D	Fuel gauge shall indicate the total fuel level in the tank	SG2	UFR18_8
H5	D	Low fuel level indicator shall activate the warning when there's low fuel level in the tank	SG3	UFR18_4

5.2 Functional Safety Concept (Pt 3, Cl 8)

The goal of this section is to collect and or provide evidence that FSRs are derived for each safety goal. As stated in the background subsection 2.4.2, FSR describes the functionality of how to achieve the safety goal. Some of AERs of FLEDS corresponds to FSR in ISO 26262. Thus a mapping from AERs of FLEDS has been made regarding the specifications of FSR. As the system is already under production, only mapping of AERs to the standard has been made. The mapping to the standard has been conducted by following the requirements of ISO 26262 in the background subsection 2.4.2.

Table 5.3: Functional safety requirements for FLEDS

Functional Safety Requirements					
SG ID	FSR ID	FSR Description	ASIL	Allocation to architectural elements	Corresponding AER
SG1	FSR 1.1	The measured fuel level signal shall be filtered to avoid that the fuel level changes rapidly in long curves, hills and slopes.	D	COO ECU System	AER_201_9
	FSR 1.2	The input signals for estimating the total fuel level shall be of good status (good status means that the signals are in the range and correct). Considered input signals are: Fuel level, fuel rate, and parking brake applied). In case input signals are not of good status; a replacement value shall be considered.	D		AER_201_20
	FSR 1.3	The input parameters used for estimating the total fuel level shall be of good status. Input parameters that are considered: sensor parameter and tanks parameters. In case input parameters are not of good status; a replacement value shall be considered and the status should not changed.	D		AER_201_21
	FSR 1.4	Fuel level estimation should use a feedback gain that shall be adjusted in a way that will not result in a deviation of more than or less than 5% from the actual volume in the tank when there's erroneous or unavailable input signals or parameters.	D		AER_201_20 & AER_201_42
SG2	FSR 2	Total fuel level should be the output of a filter that includes information from both fuel level and fuel rate in order to be able to provide the total fuel level even when one of the signals is not available or faulty.	D	COO ECU system	AER_201_11
SG3	FSR 3.1	The low fuel level warning shall warn one time when the estimated fuel level reaches below a limit of the measurable volume in the tank. The level should be 10% of tank sizes equal or below 900 liters and 7% for tank sizes larger than 900 liters.	D	COO ECU system	AER_202_2
	FSR 3.2	Signal status for the total fuel level shall be checked before checking if the low limit of the tank has been reached or not. In case the total fuel level has a bad status, the warning shall not be triggered.	D		AER_202_4

5.3 Specification of the Technical Safety Requirements (Pt 4, Cl 6)

The goal of this section is to collect and or provide evidence that TSRs are derived for each FSR. As stated in the background subsection 2.4.3, TSR describes how to implement FSR in hardware or software. Based on what have been found for FLEDS, only one requirement from ISO 26262 regarding the specification of TSR have been followed as the system is already under production and we can't specify new TSR rather than a mapping of AERs of FLEDS to TSR concept is conducted. The requirement that has been followed is stated as:

- 6.4.1.1 *The technical safety requirements shall be defined with respect to the functional safety concept as well as the elementary architectural presumptions of the item.*

Some AERs that corresponds to TSRs were not documented in AERs documents but they were implemented in the system. Thus, no mapping were found and as a result, the corresponding *AER field* in Table 5.4 is filled with no requirement about this TSR is specified in the requirement document. Table 5.4, shows the technical safety requirements for the system.

Table 5.4: Technical safety requirements for FLEDS

Technical Safety Requirement				
FSR ID	TSR ID	TSR Description	Allocation	Corresponding AER
FSR 1.1	TSR 1	Filtering of the measured fuel level should be implemented with a software functional unit. The filtering should be implemented using a low pass filter through the following equation.	Allocated in the MIDD layer of COO ECU system.	AER_201_9
FSR 1.2	TSR 2	Checking and replacement of the input signals should be implemented with a software functional unit.	Allocated in the APPL of the COO ECU system.	No requirement about this TSR specified in the requirement doc.
FSR 1.3	TSR 3	Checking and replacement of the input parameters should be implemented with a software functional unit.	Allocated in the APPL of the COO ECU system.	No requirement about this TSR specified in the requirement doc.
FSR 1.4	TSR 4	The adjustment of the feedback gain should be implemented with a software functional unit.	Allocated in AE201 in APPL layer in COO ECU system.	No requirement about this TSR specified in the requirement doc.
FSR 2	TSR 5	The filter should be implemented with a kalman algorithm given by the following equation with a feedback gain K=1.0786 * 10^-5	Allocated in AE201 in APPL layer in COO ECU system.	AER_201_11
FSR 3.1	TSR 6	Low fuel level warning should be implemented with a software functional unit.	Allocated in AE202 in APPL layer in COO ECU system.	No requirement about this TSR specified in the requirement doc.
FSR 3.2	TSR 7	Signal status checking should be implemented with a software functional unit	Allocated in AE202 in APPL layer in COO ECU system.	No requirement about this TSR specified in the requirement doc.

5.4 System Design (Pt 4, Cl 7)

As stated in the background subsection, the system design shall be in compliance with FSRs and TSRs. Therefore, this section aims to collect and or provide evidence about that. In order to achieve the goal of this section, we need first to present the system design as well as the software design. Secondly, we collect and provide evidence about the analysis on the system design to identify the causes and effects of the systematic failures. Thereafter, evidence about the verification of both the system and the software design is presented. In Scania, the design of the system was presented as simulink models. Due to the complexity of the models, the author of this thesis has modelled the system using SysML modelling language [32] in order to simplify the design and extract the information that are needed only for variant 1 of the system. COO ECU system is the main ECU that is responsible for fuel level display system functions. COO ECU system requires three inputs. The inputs are as follows:

- Fuel level signal from fuel level sensor
- Fuel rate signal from EMS ECU system
- Parking brake status from parking brake switch

The communication among systems components is as follows:

- Communication between fuel level sensor and COO ECU system is
- established through cable
- The communication between parking brake switch and COO ECU system is established through cable as well
- The communication between COO and ICL ECU system is established through SAE-J1939 connector cable using yellow CAN
- The communication between COO and ICL ECU system is established through SAE-J1939 connector cable using yellow CAN

The power supply is fed to different ECUs through cables. More information about the internal functions for the ECUs is presented in the next sections.

5.4.1 SESAMM

Scania has developed an electrical system which can be used on both trucks and buses with minor modifications. SESAMM system is composed of a number of ECU systems as shown in Figure 5.1, that are responsible for different services in the vehicle. Since the thesis is about FLEDS, thus only FLEDS related ECUs are considered. Related ECU systems of FLEDS are: EMS ECU system, ICL ECU system, and COO ECU system. EMS ECU system is responsible for all the functions that are related to the engine, ICL ECU system is responsible for all the functions that are used for displaying indications to the driver such as warning lamps, whereas COO ECU system contains all the functions that do not obviously belong to any other ECU system. The main focus of this thesis is *COO ECU* system since it has all the functions related to fuel level display system. There is a relation between EMS and COO ECU systems in which EMS send a signal to COO system and thus a relation is presented between these two ECU systems. Another relation exists between COO and ICL ECU systems, in which COO provides interfaces that are presented as indications to the driver through ICL ECU system. COO ECU System is composed of COO ECU Hardware and COO ECU Software, whereas it's aggregated by other components such as fuel sensor and PBS as the life time of fuel level sensor and PBS don't depend on the life time of COO ECU system. ICL ECU System is composed of ICL ECU Hardware and ICL ECU Software, whereas it's aggregated by other components such as fuel gauge and warning lamp.

The software for ECU system in SESAMM consists of the following three main layers:

MIDD layer

A layer where the information is processed into engineering quantities such as converting the voltage value via an analogue input to a specified value needed for calculations, decoding or encoding of CAN messages. The layer is also responsible for signal processing of input data such as filtering, linearisation, and out of range control. Each ECU has different software modules that compose MIDD layer.

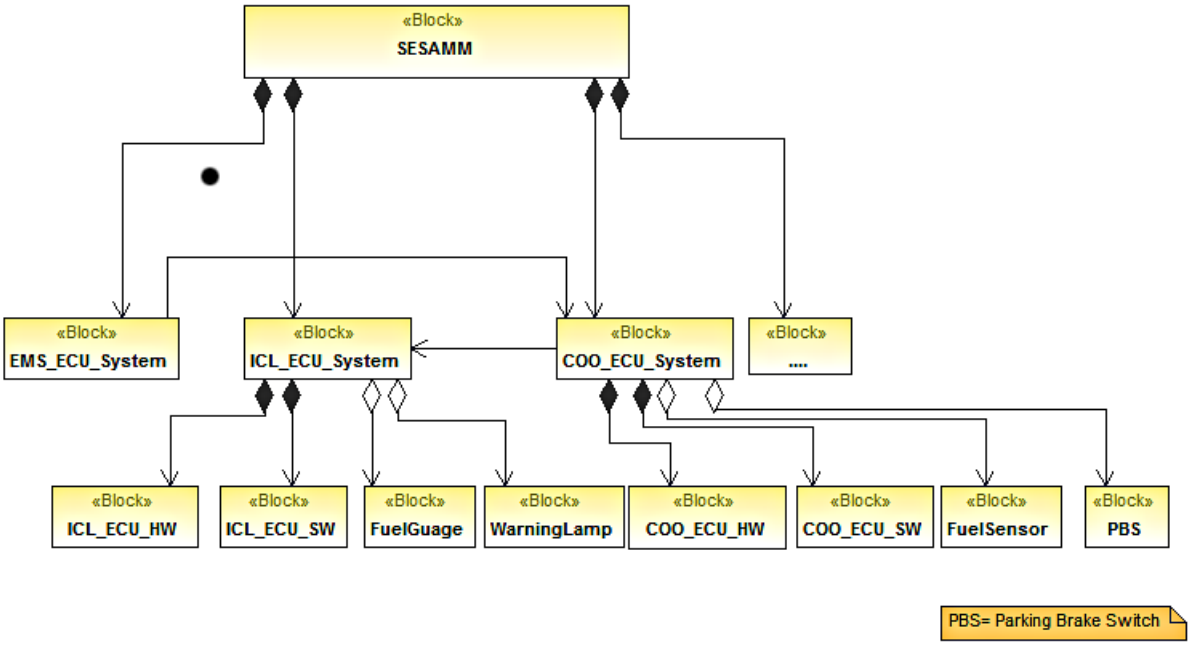


Figure 5.1: Technical view of SESAMM

APPL layer

This layer consists of software modules that handle the trucks behaviour This layer gets its input from RTDB and also delivers its results there. Each ECU has different software modules that compose the APPL layer.

RTDB layer

It's a signal layer that is used for storing the information that is to be passed between MIDD and APPL layers. For exchanging of data through the signal layer, read and write operations are used. Figure 5.2 shows the relations among those ECU software layers.

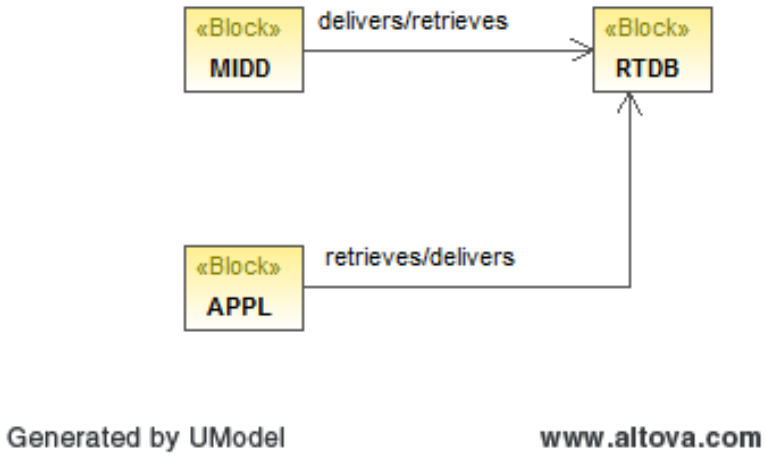


Figure 5.2: Relations among ECU software layers

5.4.2 ECU software

There are three main ECUs that are involved in FLEDS system. These system are introduced in more details in the following sections.

Instrument Cluster

ICL ECU system is responsible for indicating information to the driver. ICL ECU software is composed of MIDD and APPL layers, whereas it's aggregated by RTDB as RTDB is distributed among all ECUs. Indications related to fuel level display system that are presented to the driver are total fuel level and low fuel level warning. Total fuel level is presented on the fuel gauge of the ICL ECU system, whereas low fuel level warning is indicated by a warning lamp. APPL layer of the ECU software is composed of gauge function, warning lamp, and other functions that are out of the scope of this thesis. Figure 5.3 shows the internal functional view for the software related to the ICL.

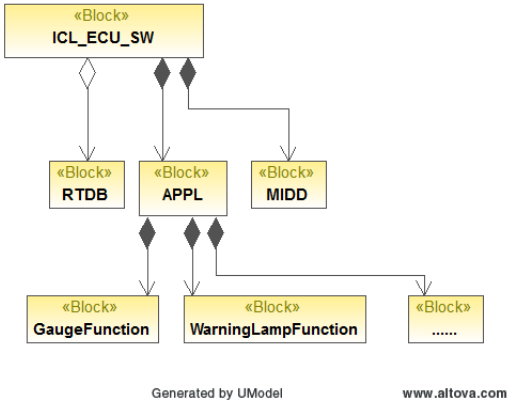


Figure 5.3: internal functional view for the software of the ICL ECU system

Coordinator

Since fuel level display functions are located in COO ECU system, thus APPL and MIDD layers have a number of functions that handle the inputs and the calculations for fuel level display. COO ECU software is composed of MIDD and APPL layers, whereas it's aggregated by RTDB as RTDB is distributed among all ECUs. MIDD layer is responsible for a number of functions that handle the inputs from sensor, PBS, and EMS ECU system. MIDD layer is composed of the following functions:

- EncodingCANMessages
- DecodingCANMessages
- MapVoltageToVolPercentage

Whereas MIDD layer is aggregated of the following function as these functions life time is independent of the MIDD layers life time. These functions are reused in other parts of the system and the parts are out of the scope of this thesis. Thus the relation between MIDD layer and these functions are represented as aggregation.

- LowPassFilter
- WritingToRTDB
- ReadingFromRTDB

More information about the purpose of each function is explained in the next few sections. APPL of the coordinator software is composed of fuel level display system and other functions. Only fuel level display system is considered in the APPL layer because other systems are out of the scope of this project. AE can contain one or more than one function. Fuel level display system is composed of two main allocation elements which are AE201 for fuel level estimation and AE202 for low fuel level warning. AE201 is composed of a number of internal functions that perform different tasks in order to estimate the total fuel level in the tank. AE201 is composed of:

- CalculateCurrentVolume
- ScaleFuelConsumption
- GainCalculation
- RefuelDetection
- KalmanAlgorithm
- ConvertToPercent

Whereas AE202 is composed of a number of internal functions in order to check if there is a low fuel level in the tank. AE202 is composed of:

- CheckSignalStatus
- WarningLevel
- SetStatus

MIDD layer receives the input signals and then writes it to RTDB in which the latter pass the signals to APPL layer. When total fuel level is calculated and low fuel level warning is checked, the values are sent from APPL to RTDB, in which the values are passed from RTDB to MIDD layer. The latter encodes the values in CAN messages that are to be sent to ICL ECU system.

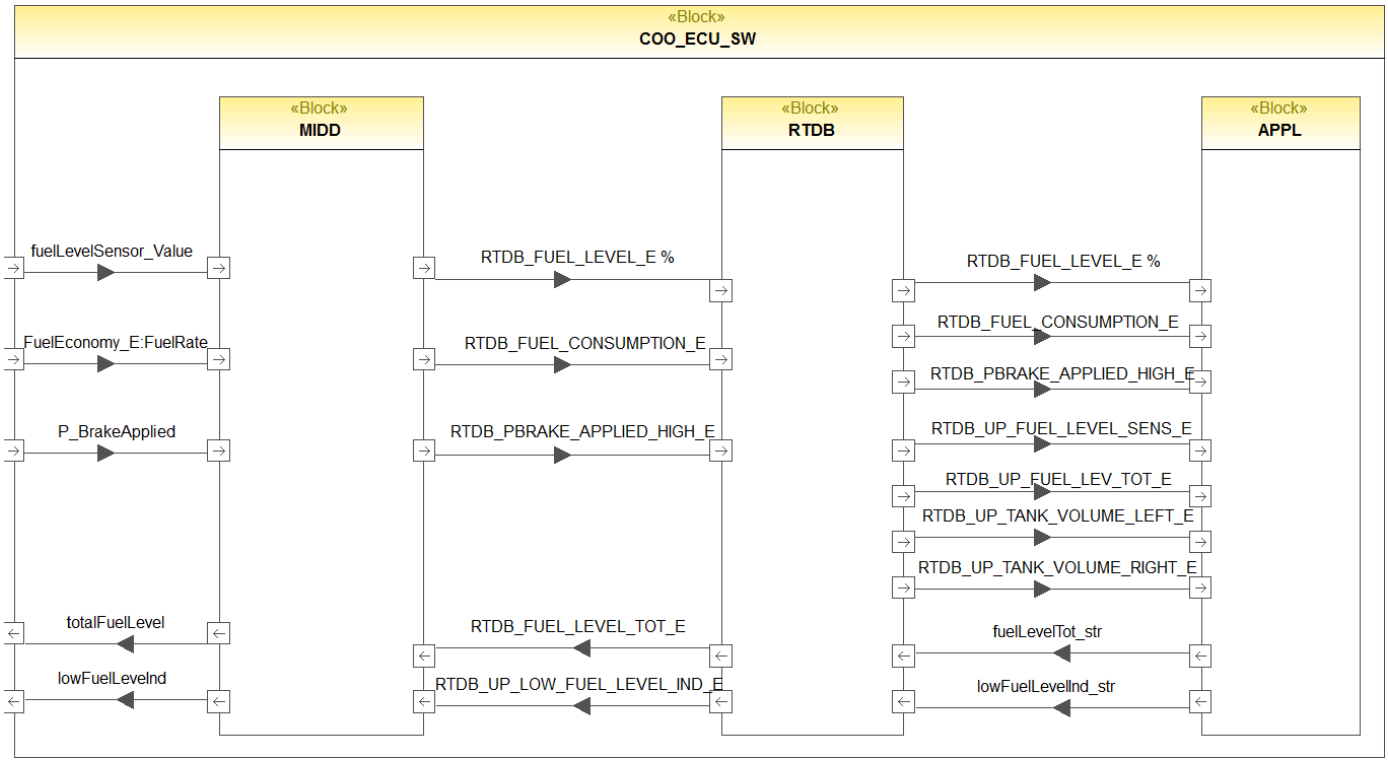


Figure 5.4: Information flow among software layers in COO ECU system

5.4.3 Software Layers in Coordinator ECU system

This section will introduce the input signals to each layer and then explains the functional components that compose each layer.

MIDD Layer

The layer is responsible for signal processing of input data such as filtering, linearisation, and out of range control as it was mentioned in the beginning of this report. MIDD layer receives input signals from different components. The input signals are as follows:

- fuelLevelSensorValue: indicates fuel level sensor signal that is received from fuel level sensor
- fuelEconomy E:FuelRate: indicates fuel rate signal that is received from EMS ECU system though CAN message
- PBrakeApplied: indicates the parking brake state that is received from parking brake switch

The layer is responsible for processing of the signals through internal functional units. Afterwards the layer passes these signals to RTDB layer. The internal functional units of MIDD layer and their purposes are as follows:

- MapVoltageToVolPercent
  This functional unit is responsible for converting the analogue input voltage into fuel level in percentage. The unit has a lookup table that maps the voltage to the corresponding level in percentage.
- LowPassFilter
  After converting the analogue input into a percentage value, the converted value is filtered by applying a low pass filter. The low pass filter is implemented by applying a specific equation. The purpose of low pass filter is to remove the high value frequencies that result when the vehicle moves uphill or downhill.
- DecodingCANMessages
  Fuel rate from EMS ECU system is received as a CAN message. In order to get the value of fuel rate, the CAN message have to be decoded. So this unit is responsible for decoding CAN messages that are received from other ECU systems.
- WritingToRTDB
  The filtered fuel level, fuel rate and parking brake applied values has to be written to RTDB. WritingToRTDB functional unit is responsible for this task.

- ReadingFromRTDB This functional unit is responsible for reading the total fuel level and low fuel level warning from RTDB.

- EncodingCANMessages
  Total fuel level and low fuel level warning has to be sent to ICL ECU system in order to be presented on the instrument cluster. EncodingCANMessages functional unit is responsible for encoding these values into messages that are sent through CAN to ICL ECU system. Figure 5.5, shows the internal functional view of MIDD layer.

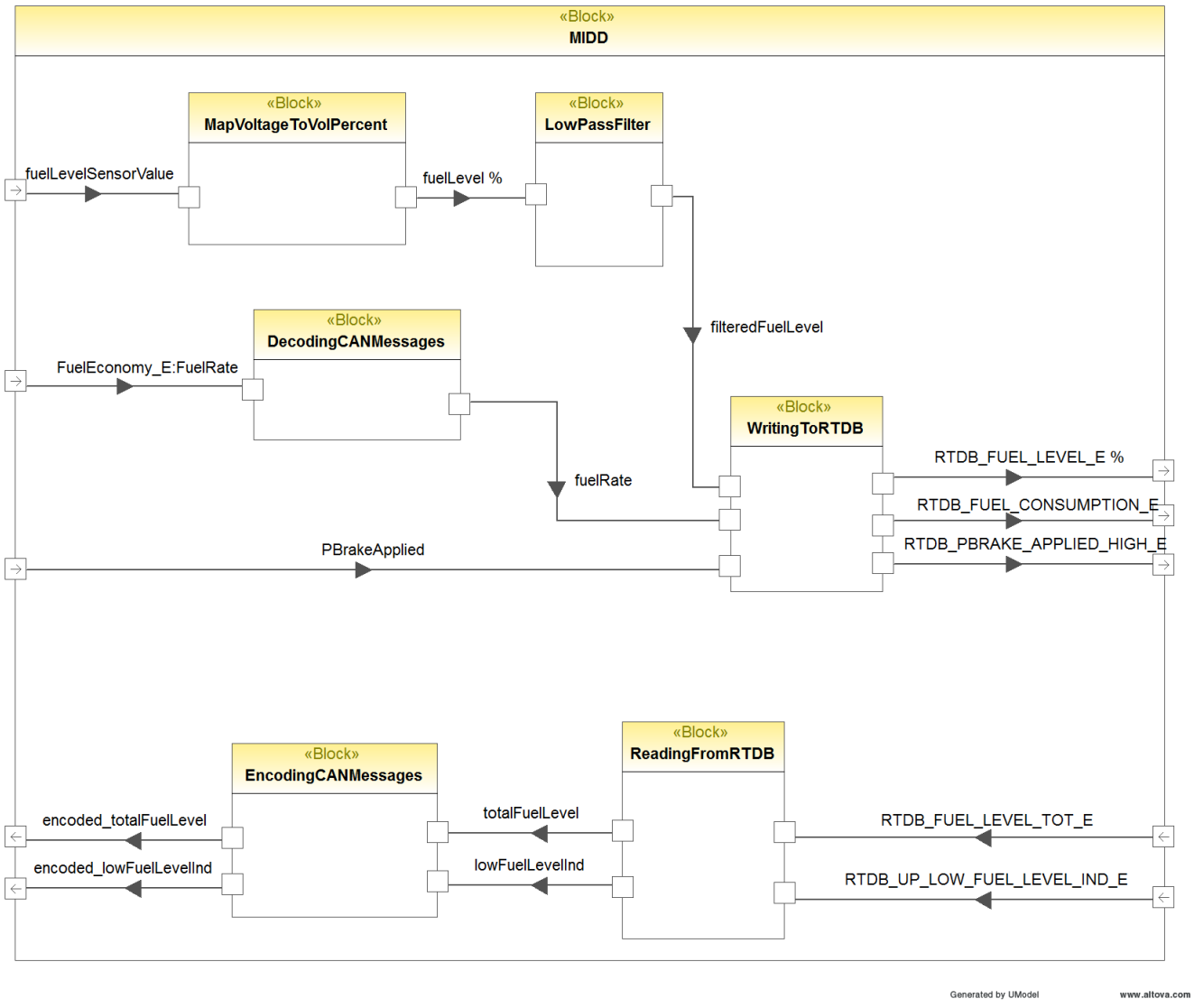


Figure 5.5: Internal functional view of MIDD layer

RTDB Layer

This is a signal layer that s used for signal storage and signal passage between MIDD and APPL layer as it was mentioned before in the beginning of this report. No information has been provided about the internal working mechanism for this layer because it's out of the scope of this project.

APPL Layer

This layer consists of software modules that handle the trucks behaviour. One of the software modules that are within the scope of this project is fuel level display system. More details about the internal functional units of this module are presented in details in the next sections.

- Fuel Level Estimation and Display System
  There are a number of requirements for fuel level display system that must be met. The system is designed in a way that makes it possible to meet its requirements. Fuel level display system is composed of two main functions; AE201 and AE202. Some functions besides these two main functions are used to do some calculations of values that are required in the calculations of the main functions. Other functions are used in order to provide safety mechanisms. There are two types of inputs that are used in fuel level display system; input signals and input parameters. Input signals are the actual signals coming from fuel level sensor, PBS, and EMS ECU system. Whereas input parameters are used to specify the sensor type used in the particular truck, tank sizes used, and which ECU system should be responsible for fuel level estimation. Input parameters are set manually by the engineers. Other functions other than the main functions are used as it was mentioned in the beginning of this section.

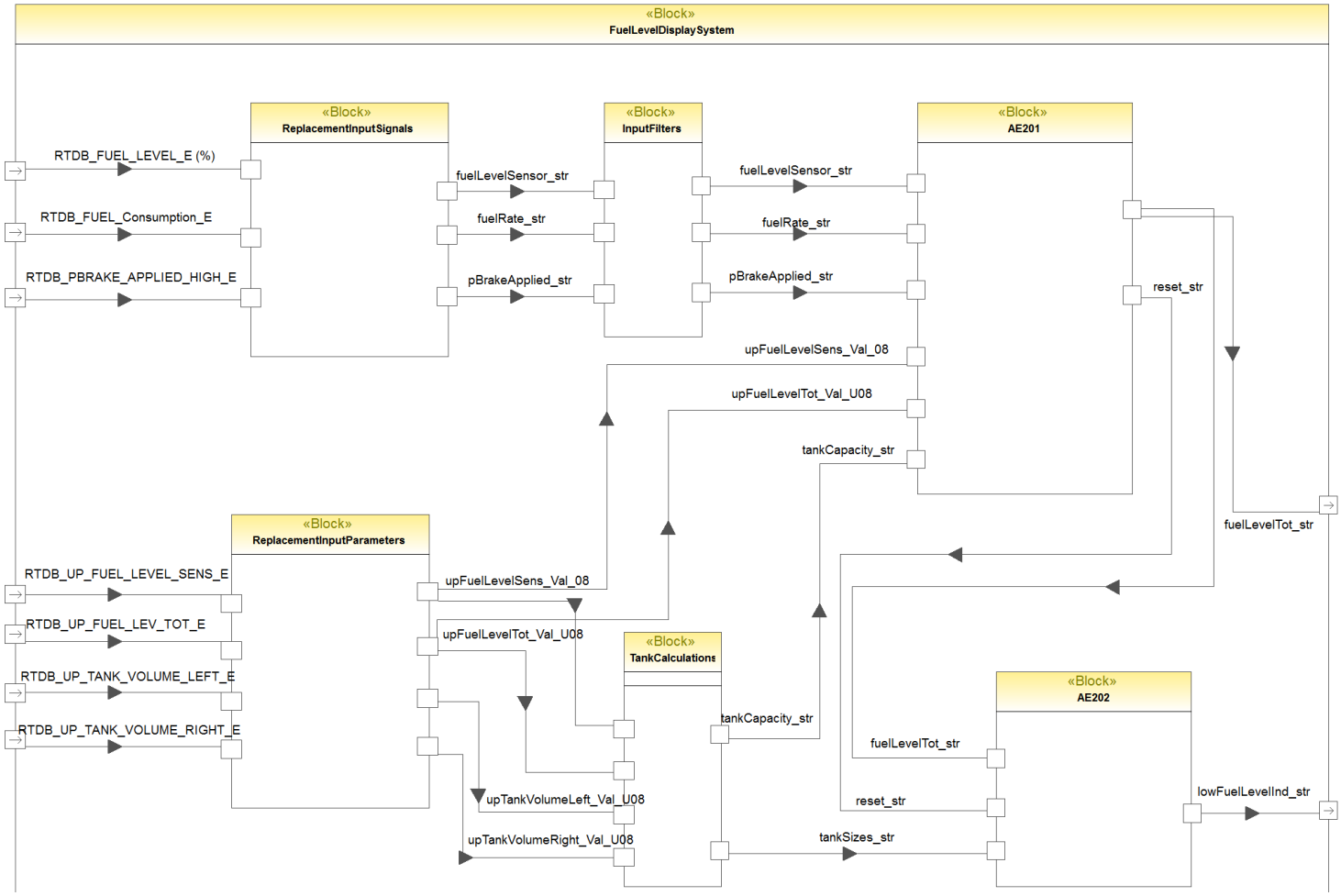


Figure 5.6: Internal functional view of fuel level display system

So the functions (functional blocks) that compose FLEDS are: *ReplacementInputSignals, ReplacementInputParameters, InputFilters, TankCalculation, AE201 and AE202*. More details about the those functional blocks of are presented in the following sections. Safety mechanisms related functions are:

ReplacementInputSignals

This is one of the safety mechanisms that are present in the system. This functional block is responsible for checking the input signals from RTDB layer. The block will check the value of the input and then assign a status for it. The status will be either Good or Not Good. The output from this block will be a string that contains the value for each input along with its status. If the status of the input is Not Good, a replacement value to be passed instead of the original value. The original status will be passed as well which means that when a value is replaced, the status Not Good is passed anyway. It receives three input signals from RTDB which are fuel level percentage, fuel rate, and parking brake applied.

ReplacementInputParameters

This is one of the safety mechanisms that are present in the system. This component will replace the faulty parameters that are coming from RTDB with safe signals in order to avoid undesired events when faulty parameters are present. The input parameters are used for configuration of the fuel level display system. The component will check the value of the input and then assign a status for it. The status will be either Good or Not Good. If the status of the input is Not Good, a replacement value to be passed instead of the original value. It receives four configuration parameters as explained below:

- RTDB UP FUEL LEVEL SENS E
  Based on the value of this parameter, the fuel level sensor that is placed in the vehicle is defined. The values range from 10 to 20 where 20 mean that the fuel type is gas and there will be no sensor for measuring the fuel level. Thus receiving gas level from EMS unit (This case is out of the scope of this project).
- RTDB UP FUEL LEVEL TOT E
  Based on the value of this parameter, the total fuel level will be estimated by one of the ECU units in the system. If the value is 10 then the total fuel level is estimated by COO, otherwise if the value of this parameter is 20 then the total fuel level is

estimated by Bus Chassis System (BCS) which is out of the scope of this project. Figure 9 shows the configuration parameters for fuelLevelSensorParam and fuelLevelTotalParam.

- RTDB UP TANK VOLUME LEFT E
  This parameter indicates the fuel tank volume on the left side in liter unit. The parameters values range from 1 to 35, where every value correspond to a specific capacity.
- RTDB UP TANK VOLUME RIGHT E
  This parameter indicates the fuel tank volume on the right side in litre unit. The parameters values range from 1 to 36, where every value correspond to a specific capacity.

Other functions other than safety mechanism functions are:

InputFilters

This block is responsible for evaluating the digital input from PBS. It checks whether the PBS is applied or not. Other inputs to this block are just passed with no processing.

TankCalculation

Tank capacity is necessary in fuel level estimation since it's used in kalman filter algorithm in order to calculate total fuel level. Tank sizes value is also necessary since it's used to set different trigger levels for low fuel level warning. Thus TankCalculation block is used to calculate tank capacity and tank sizes. The difference between tank capacity and size is that tank capacity means the total usable volume including the hidden volume, bulb volume, end plate volume and etc., whereas tank size is the actual size of the tank. TankCalculation functional unit consists of the following internal functional units:

- ◇ ParameterCheckLogic
  This functional unit is responsible for checking if the fuel level sensor parameter or one of the tank volume Parameters has invalid value. The output from this function is used in FindVolumes function.
- ◇ MapParameterToTankSize
  This functional unit is responsible for mapping the tank volume parameters to the actual tank size. If there is a left and a right tank then both of them shall be considered. The mapping is based on a look up table.

◇ FindVolumes
This block is used to calculate the static volume. Static volume is the total volume in the tank including the hidden volume, the bulb volume and etc. The calculation will be based on the sensor type. Each sensor has a different number of steps that are used to read the level in the tank. Fuel level sensor will not cover all the volume in the tank; instead it will cover the volume from the minimum point of the sensor until the maximum points of the sensor. Thus there are volumes that are not considered in the measurement of the sensor and it should be considered in the estimation algorithm. This function will handle the volumes that are not considered by the sensor. The output from this block will contain all the calculated different volumes (hidden, bulb, end plate, and etc).

◇ AddLevels
This block is responsible for accumulating all the volumes received from FindLevels block. The result of accumulation is the tank capacity. The output is a string that contains both tank capacity and wrong tank parameter setting.

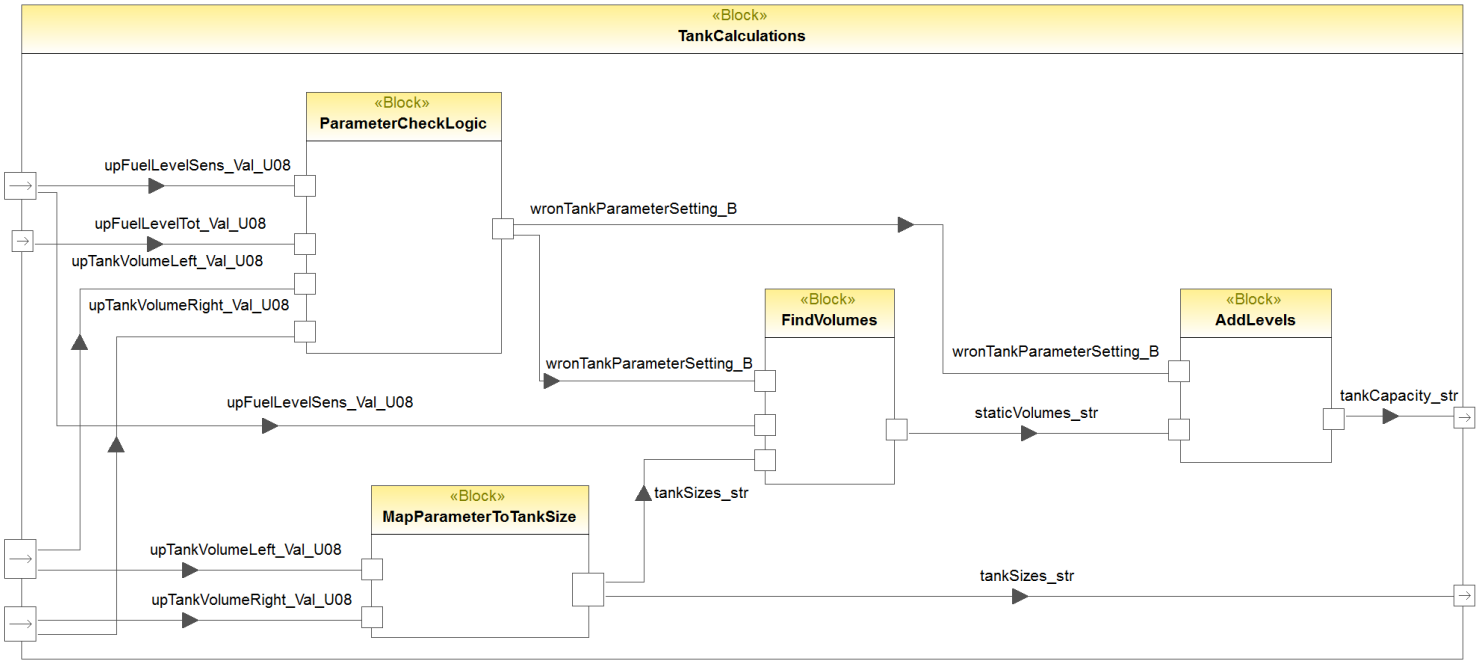


Figure 5.7: Internal functional view of TankCalculation

AE201

This functional unit is responsible for calculating the total fuel level that is presented on the fuel gauge. AE201 consists of a number of functional units in order to estimate the total fuel level. Each signal that ends with *str* means that it contains a value (Val) and a status (SS). Figure 5.8, show the internal functional view of AE201.

Internal blocks of AE201 are:

- CalculateCurrentFuelVolume
  Fuel level received from RTDB layer is just the level measured by the sensor and converted to percentage. The level measured by the sensor represents only what is in the range of the sensor. There are hidden volumes that are not measured by the sensor. Thus it cant be used directly in kalman filter algorithm. Fuel volume is required in order to estimate the total fuel level. Two inputs are needed in order to calculate the current fuel volume. These inputs are fuel level sensor value and tank capacity.

- ScaleFuelConsumption
  The fuel rate received from EMS ECU system is received as liter per hour. This block takes the fuel rate value, scales it, and then converts it to cubic meter per hour in order to be used in the algorithm. The status for fuel rate is checked. In case fuel rate status is Not Good, kalman gain is changed in a way that makes it possible to not be affected by the bad status of fuel rate. More details about this are explained further in this report.

- RefuelDetection
  One of the requirements for fuel level display system is to detect the refuel made by the driver. The purpose of this requirement is to re-estimate the total fuel level when the tank is refueled. In order to be able to check if there is a refuel, a parking brake signal is necessary to be checked, and a fuel level in the tank must be monitored. The fuel level must be monitored in order to see whether there is at least 30% increase in the fuel level or not. The reason why parking brake signal to be checked is that because the driver usually applies the parking brake before he starts to refill the tank. Therefore refuel detection function is needed in order to detect the refuel action. Two inputs are required to this block as it's obvious from figure 12. If a refuel is detected, a reset string is set to true. When reset string is true, the kalman algorithm and low fuel level warning are reset too.

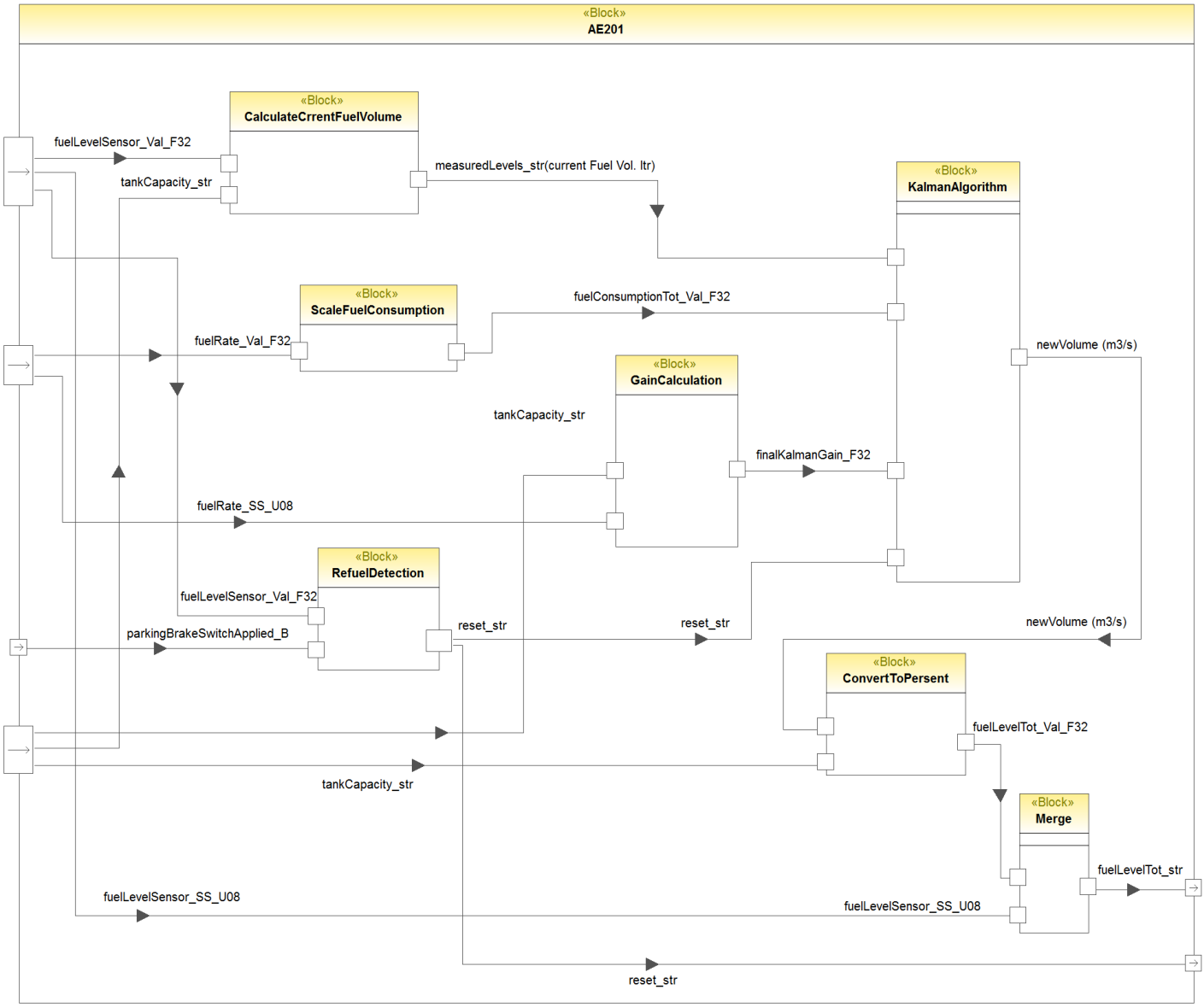


Figure 5.8: Internal functional view of AE201

- GainCalculation
  Gain feedback is used in kalman algorithm to estimate the total fuel level. The value of gain feedback is very essential and it affects the result of the algorithm. Thus it's very important to write a function that takes care of gain in case some of the inputs to kalman algorithm are not of Good Status or Not Available. GainCalculation functional block checks fuel rate status and tank size parameters. If there's an error in fuel rate or if tank size parameters are not set, then feedback gain is replaced with a constant value in order to make the algorithm continues to calculate total fuel level by only using fuel level from the sensor as an input. Figure 5.9, shows internal functional view of GainCalculation.

- KalmanAlgorithm
  It's an algorithm that is used to estimate the total fuel level in the tank. It uses the fuel volume and fuel consumption in the calculations. The output from the algorithm is the volume in cubic meter per second.

- ConvertToPercent
  The output from kalman algorithm is in cubic meter per hour. Since total fuel level is indicated on the fuel gauge as a percentage between 0 and 100, thus a conversion from cubic meter per hour to percentage is essential. The output from this function is the total fuel level value in percentage.

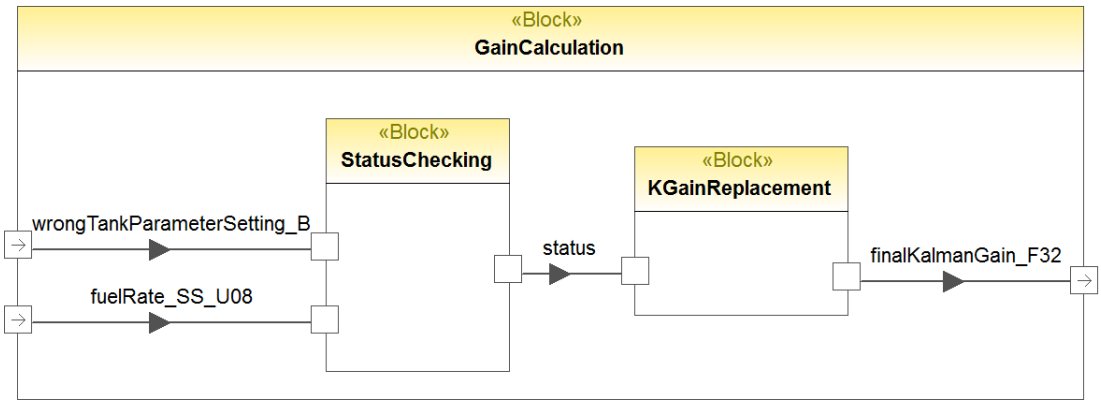


Figure 5.9: Internal functional view of GainCalculation

AE202

The low fuel level warning is used to warn if the estimated fuel level is below a predetermined level in the tank. The limit is 10% for tank sizes below 900 litres and 7% for larger tanks. Low fuel level warning function consists of internal functions to achieve the required functionality. Internal functions and their purposes are as follows:

- CheckSignalStatus
  CheckSignalStatus function is used to check the status of total fuel level. Before checking if low limit has been reached or not, total fuel level status is checked. In case total fuel level status is Not Good, total fuel level is set to a value that doesn't trigger a low level warning.

- WarningLevel
  WarningLevel function is used to check if low limit has been reached or not. In case fuel level total status is Good then an input from tank size is used to check the size of the tank in order to be able to know the limit that has to be reached in order to indicate if low level has been reached or not.

- SetStatus
  SetStatus functional block is used to either set or reset the low level warning. If a refuel of the tank is detected then the warning is not set, otherwise the warning is set.

After presenting the system and the software architectural design, the next step is to analyse the design by following 7.4.3 requirement. The requirement states that:

- 7.4.3.1 *Safety analyses on the system design to identify the causes of systematic failures and the effects of systematic faults shall be applied in accordance with 5.5 [17].*

Table 5.5: System design analysis methods

Methods		ASIL			
		A	B	C	D
1	Deductive analysis[a]	o	+	++	++
2	Inductive analysis[b]	++	++	++	++

[a] Deductive analysis methods include FTA, reliability block diagrams, Ishikawa diagram.

[b] Inductive analysis methods include FMEA, ETA, Markov modelling.

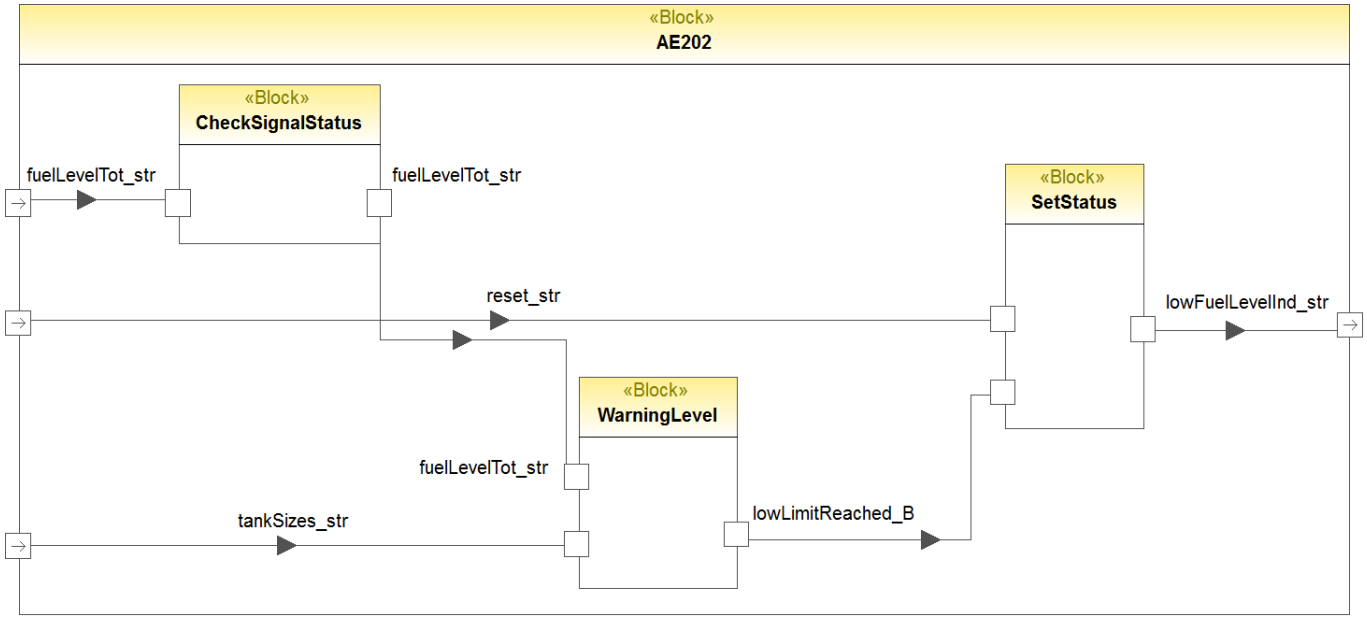


Figure 5.10: Internal functional view of AE202

The requirement suggests the use of inductive or deductive analysis methods for the assessment of the design. The design has been assessed in Scania by using FMEA. The next set of tables provides FMEA that were done by Scania. The tables provides sufficient information about different failure modes and their effects. The tables are not completely compatible with the sheets presented in the background subsection 2.3.1. However, the analysis done by Scania shows that the company made efforts to consider the consequences as well as provided mechanisms in the design to overcome the possible failures that may occur. .

Table 5.6: FMEA for the system made at Scania-part 1

FMEA						
Item		General failure mode	Interpretation	Vehicle effect	Severity	Preferred fail state
fuelLevelSensor	Timing	Late	Fuel level cannot be measured.	The total fuel level has bad status	L	
		Early	No problem if fuel level is received early			
	Occurrence	Omission	Fuel level sensor can give values it is not suppose to. This is handled in the Value cases.			
		Commission	Fuel level sensor can give values it is not suppose to. This is handled in the Value cases.			
	Value	Implausible	Fuel level sensor value cannot be used if it is outside its boundaries, DTC is created.	The totalFuelLevel has bad status.	L	X
		Plausible but incorrect	Fuel level sensor value is trusted.	Fuel level is estimated as valid but the value is incorrect.	H	
parkingBrakeApplied	Timing	Late	Parking brake is not applied.	A parking of vehicle can not be detected until signal goes high. Refuel can only be detected when the vehicle is parked.	L	X
		Early	No problem if parking brake signal is received early.			
	Occurrence	Omission	P-brake signal can indicate the opposite of the true parking brake state. This is handled by Value – Plausible but incorrect.			
		Commission	P-brake signal can indicate the opposite of the true parking brake state. This is handled by Value – Plausible but incorrect.			
	Value	Implausible	Not applicable, signal only has two states.			
		Plausible but incorrect	Parking brake signal is trusted.	Vehicle can be considered to be parked while driving and refuel detection can be active. Resulting in false behavior of totalFuelLevel.	M	
fuelRate	Timing	Late or Early	Large timing differences in fuel consumption affects fuel level estimation algorithm.	totalFuelLevel can behave strange if timing between fuel consumption in engine and fuel level sensor	L	

Table 5.7: FMEA for the system made at Scania-part 2

				changes differ.		
	Occurrence	Omission	Signal is not available.	Fuel level can still be displayed but is based on only fuel level sensor.	L	
		Commission	Case covered by Value – Plausible by incorrect.			
	Value	Implausible	If value is implausible it's limited.	Implausible values are only limited to highest or lowest plausible value and algorithm continues. Fuel level sensor will support the estimation of totalFuelLevel.	L	
		Plausible but incorrect	fuelRate is correct if its status is good.	Incorrect values affects the estimation but is supported by fuel level sensor.	L	X
fuelLevelSensorParam	Value	Implausible	No sensor present.	No fuel level estimate. totalFuelLevel gets bad status.	L	X
		Plausible but incorrect	Sensor present.	Wrong sensor mapping achieved and estimated fuel level is false. At some fuel levels electrical DTCs are set.	M	
fuelLevelTotalParam	Value	Implausible	Fuel level estimation off.	No fuel level estimate. totalFuelLevel gets bad status.	L	X
		Plausible but incorrect	Fuel level estimation running.	Wrong main variant is set, see AER_201_1. For some combinations with fuelLevelSensorParam a fuel level estimate will be achieved but incorrect.	M	
fuelTankSizeLeft	Value	Implausible	Tank volume not set.	No fuel level estimate. totalFuelLevel gets bad	L	X

Table 5.8: FMEA for the system made at Scania-part 3

				status.		
		Plausible but incorrect	Tank volume not set correct.	Incorrect values affect the estimate but the estimate is supported by fuel level sensor.	L	
fuelTankSizeRight		See fuelTankSizeLeft				
totalFuelLevel	Timing	Timing is not relevant; it doesn't matter if the signal is early or late. The value and status are the relevant cases.				
	Occurrence	Omission	Signal is not available.	If no fuel level is available no low fuel level warning will be set.	M	
		Commission	This case is covered by Value-Implausible.			
	Value	Implausible	totalFuelLevel is interpreted as correct if its status is good.	The implausible fuel level will be used as if it is correct.	M	X
		Plausible but incorrect	totalFuelLevel is interpreted as correct if its status is good.	The incorrect fuel level will be used as if it is correct.	M	
lowFuelLevelWarningParam	Timing	Not relevant, parameter either exists or not. As long as the parameter is implemented it will be created.				
	Occurrence	Not relevant, parameter either exists or not. Implausible and plausible but correct covers the relevant cases for occurrence.				
	Value	Implausible	Low fuel level warning is not active since the parameter is not set to a supported value.	lowFuelLevelWarning will never be set	M	
		Plausible but incorrect	Low fuel level warning should be active or not active depending on value.	lowFuelLevelWarning will be set according to AER_202_2	L	X

However in order to be able to know if there were any safety mechanisms for particular causes, an analysis of the design with respect to the causes was essential. Moreover, the standard requires an an assessment of the design as presented in the following requirement:

- 7.4.3.1 *Safety analyses on the system design to identify the causes of systematic failures and the effects of systematic faults shall be applied in accordance with Table 5.5*

Thus, the design has been analysed using FTA as it was recommended by the standard. FTA is based on the hazards that have been identified in HAZOP that is in Table 5.1.
Based on hazard and operability study (HAZOP) and FMEA, possible deviations that can occur in the system have been defined. Deviations that have been defined and considered are as follows:

- Fuel gauge indicates higher fuel level than actual fuel level in the tank.
- Fuel gauge indicates lower fuel level than actual fuel level in the tank.
- Fuel gauge indicates no fuel level when it should not.
- Fuel level warning displayed when it should not.
- Fuel level warning not displayed when it should.

For each of the deviations presented above, a fault tree is presented in order to investigate the possible causes of each deviation. Every fault tree shall analyse one and only one deviation or undesired event. Fault tree analyses and its description for the considered deviations are as follows:

- Fuel gauge indicates higher fuel level than actual fuel level in the tank
  The possible causes for this deviation can be either a mechanical fault in fuel gauge, Bug in gauge function in ICL ECU system that lead to setting wrong steps for fuel gauge, or erroneous fuel estimation by kalman filter. Kalman filter depends on fuel rate and fuel level in order to estimate the total fuel level in the tank. Thus erroneous fuel estimation by kalman filter can be caused when any of these inputs has an erroneous value. Fuel level is measured by fuel level sensor, then low pass filtered, and afterwards used with tank capacity in order to calculate the usable fuel level in the tank. Thus possible causes of erroneous fuel level can be either by erroneous fuel level sensor value, erroneous filtering of fuel level value , or erroneous calculation of tank capacity in which the latter is caused by either erroneous tank parameter settings or erroneous sensor parameter setting. Fuel rate is calculated in EMS ECU system. Thus erroneous fuel rate results from fault in fuel rate calculations by EMS ECU system. Figure 5.11, shows fault tree for this deviation.

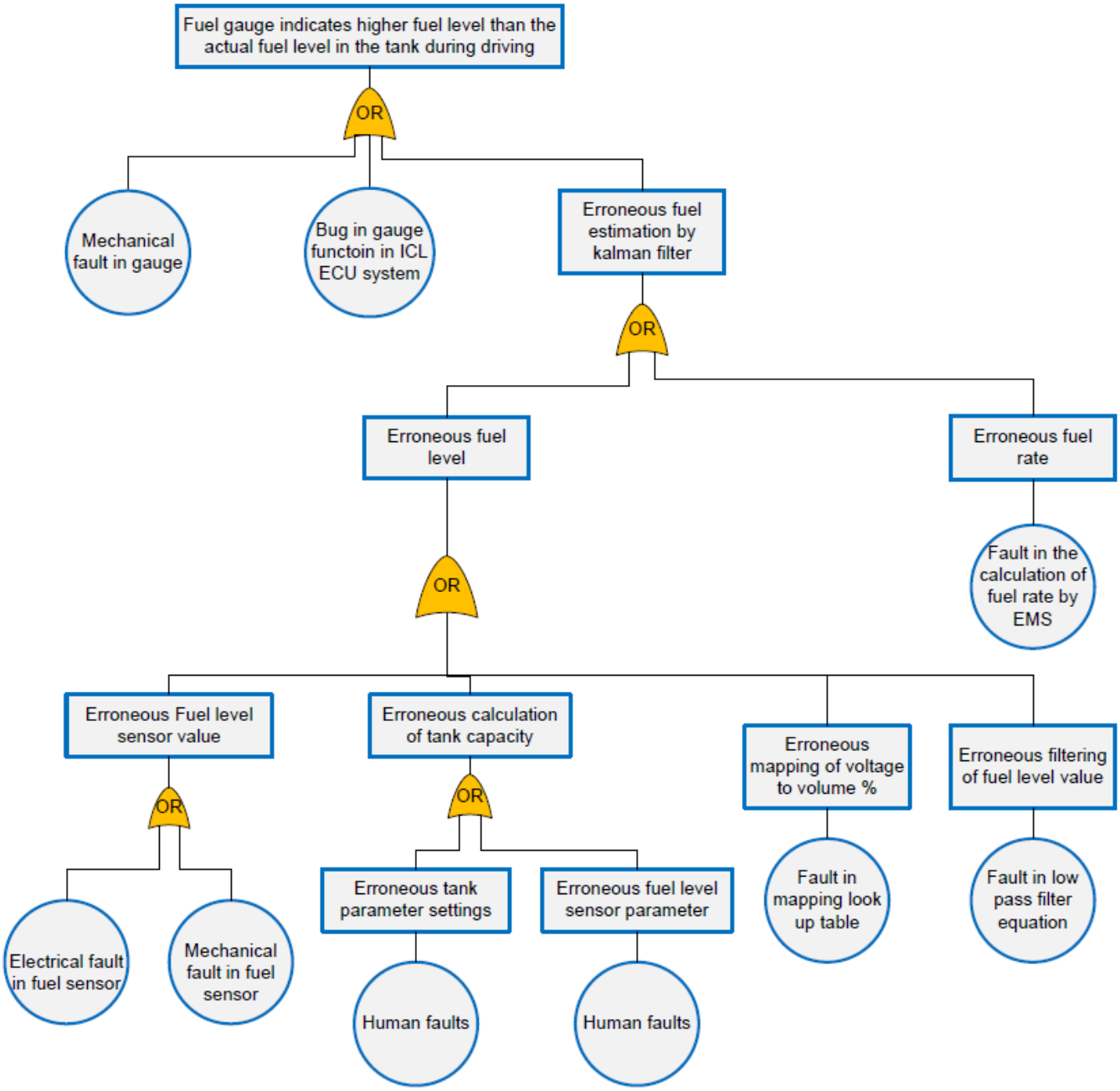


Figure 5.11: Fault tree for the deviation "Fuel gauge indicates higher fuel level than the actual fuel level in the tank"

- Fuel gauge indicates lower fuel level than actual fuel level in the tank
  The possible causes for this deviation are the same as the causes for deviation Fuel gauge indicates higher fuel level than actual fuel level in the tank. Figure 5.12, shows the fault tree for this deviation.

- Fuel gauge indicates no fuel level when it should not
  The possible causes for this deviation can be a mechanical fault in fuel gauge, a hardware fault in ICL ECU system, a bug in gauge function in ICL ECU system, or there's no total fuel level received by ICL ECU system. The possible causes for no total fuel level received by ICL ECU system can be a communication problem between COO and ICL ECU system or problem in COO ECU system. The communication problem can be either because of a cut in the communication cable or a lost CAN message that contains the total fuel level, in which the latter is caused by a Fault in the CAN bus. Problem in COO ECU system can be caused by a hardware fault in COO or when no power supply is fed to COO, in which the latter is caused by a fault in power supply or a fault in power supply cable between COO and the power supply source. Figure 5.13, shows the fault tree for this deviation.

- Fuel level warning displayed when it should not
  The possible causes for this deviation can be either a bug in warning lamp function in ICL ECU system or an erroneous output from low fuel level warning function in COO ECU system. Erroneous output from low fuel level warning is caused by erroneous fuel estimation by kalman filter or erroneous value of tank sizes. Erroneous fuel estimation by kalman filter is caused by erroneous fuel level or erroneous fuel rate in which the latter is caused by a fault in the calculations of fuel rate by EMS. Erroneous fuel level is caused by erroneous fuel level sensor value, erroneous mapping of voltage to volume percentage, erroneous filtering of fuel level value, or erroneous calculations of tank capacity. Erroneous calculations of tank capacity are caused by erroneous tank parameter settings or erroneous sensor parameter in which both are caused by human faults. Erroneous filtering of fuel level value is caused by fault in low pass filter equation. Erroneous mapping of voltage to volume in percentage is caused by fault in mapping look up table. Whereas erroneous fuel level sensor value is caused by either electrical or mechanical fault in fuel sensor. Figure 5.14, shows the fault tree for this deviation.

- Fuel level warning not displayed when it should
  The possible causes for this deviation can be a communication problem between COO and ICL ECU systems, a fault in warning lamp (burned), a bug in warning lamp function in ICL ECU system, erroneous output from low fuel level warning function in COO ECU

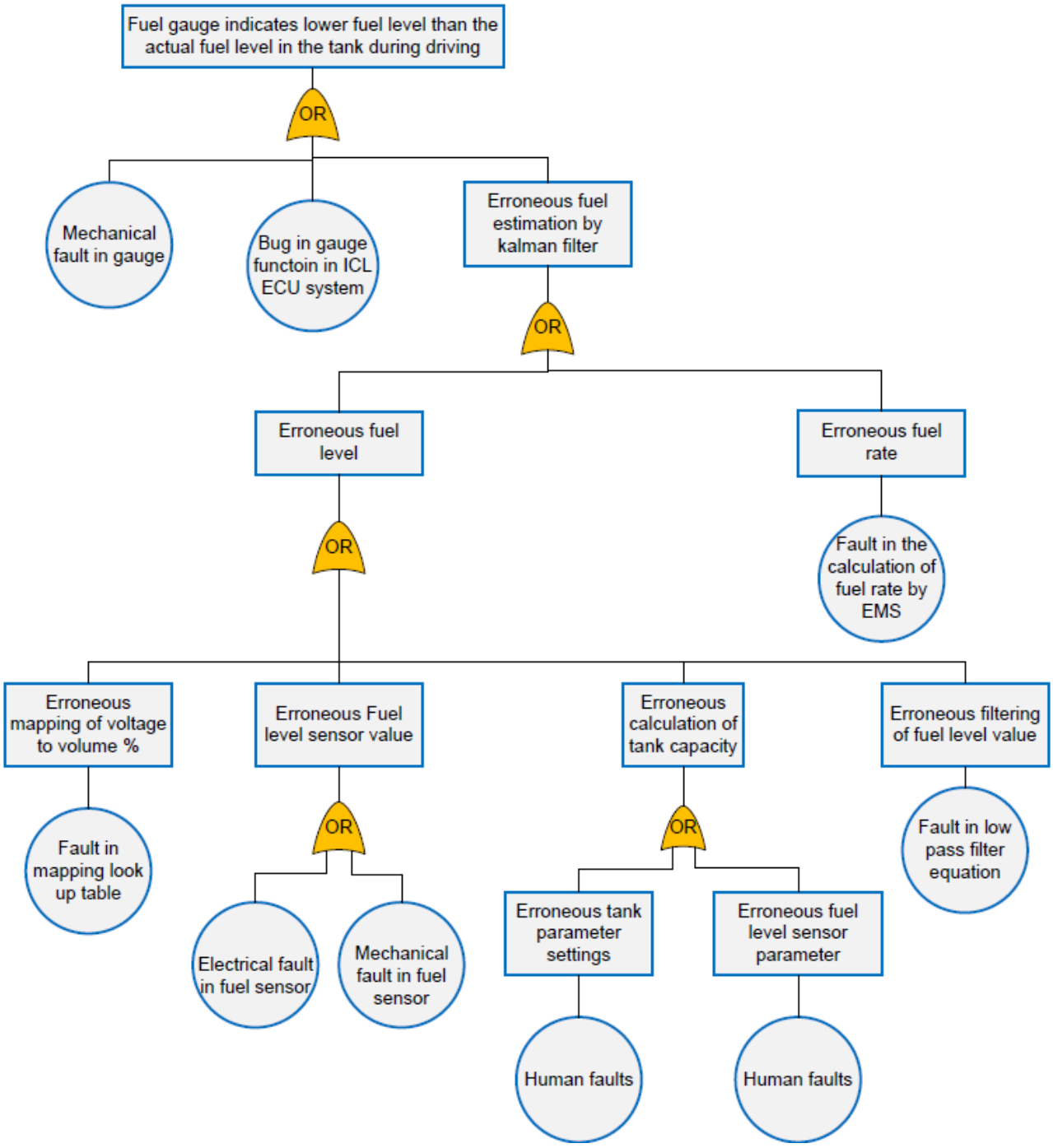


Figure 5.12: Fault tree for the deviation "Fuel gauge indicates lower fuel level than the actual fuel level in the tank"

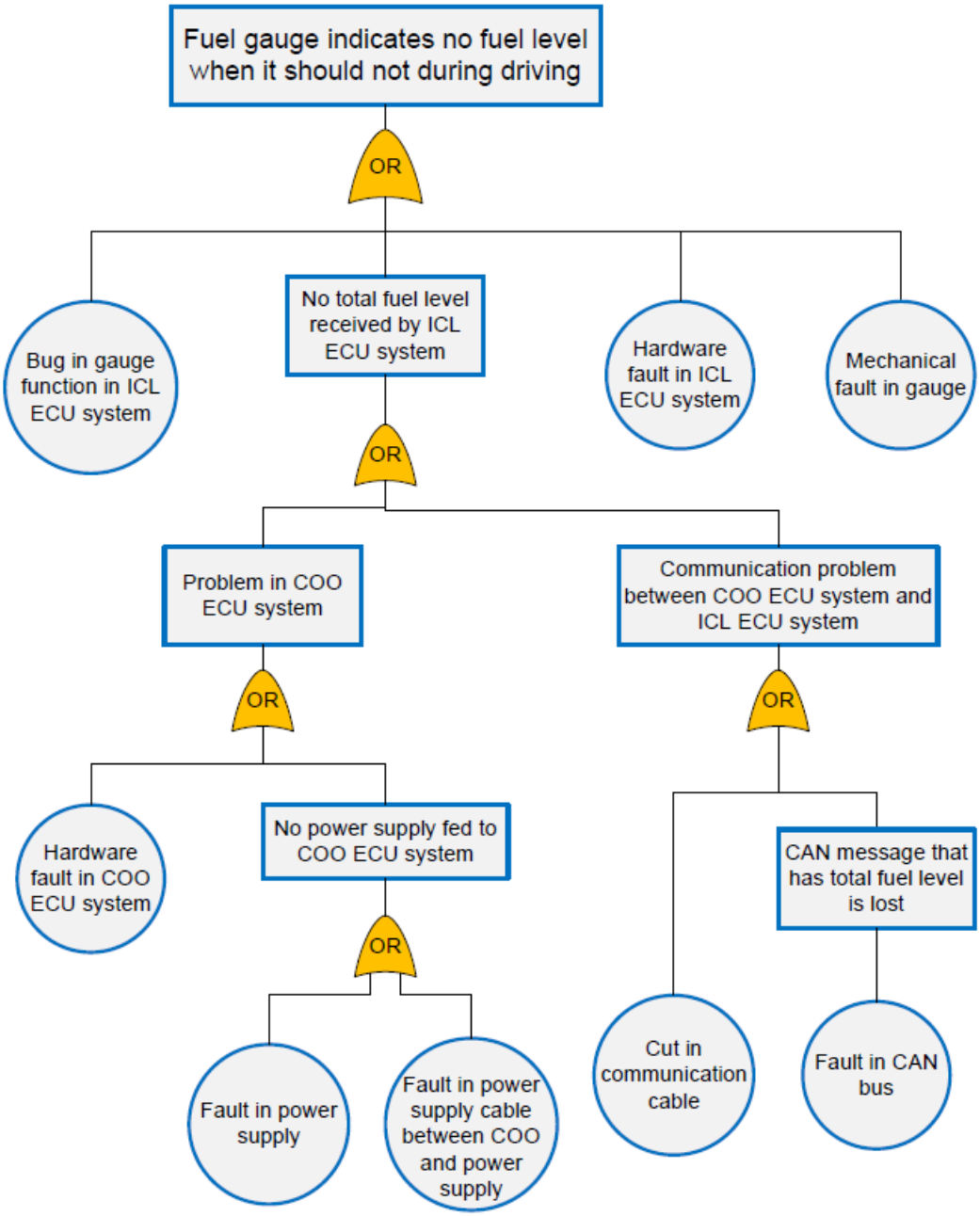


Figure 5.13: Fault tree for the deviation "Fault tree for the deviation Fuel gauge indicates no fuel level when it should not"

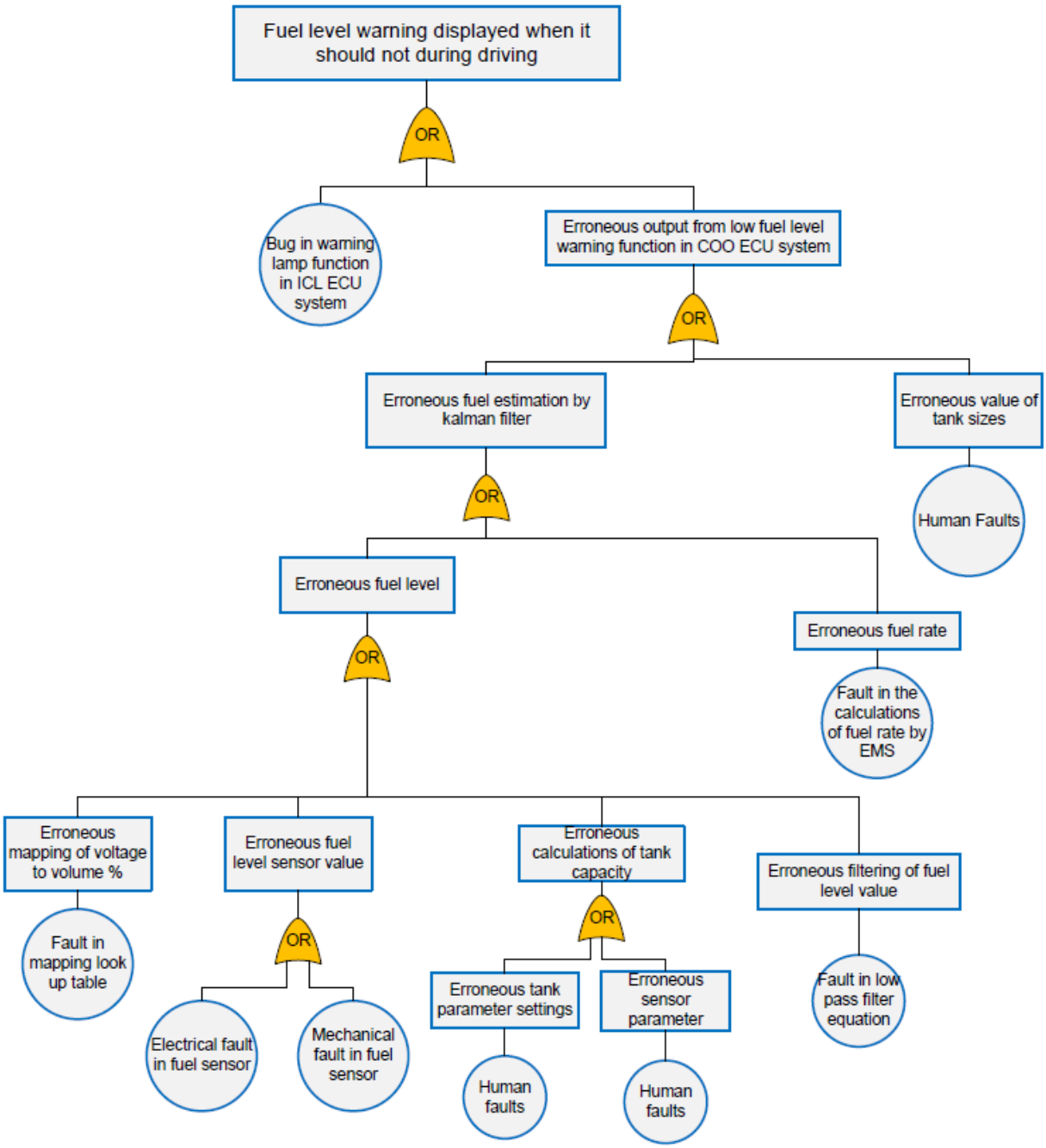


Figure 5.14: Fault tree for the deviation "Fuel level warning displayed when it should not"

system. The communication problem between COO and ICL ECU system can be either a cut in the communication cable or a lost CAN message that contains the activation of the warning in which the latter is caused by a fault in the CAN bus. Erroneous output from low fuel level warning function in COO ECU system is caused by erroneous fuel estimation by kalman filter or erroneous value of tank sizes. Erroneous fuel estimation by kalman filter is caused by erroneous fuel level or erroneous fuel rate in which the latter is caused by a fault in the calculations of fuel rate by EMS. Erroneous fuel level is caused by erroneous fuel level sensor value, erroneous mapping of voltage to volume percentage, erroneous filtering of fuel level value, or erroneous calculations of tank capacity. Erroneous calculations of tank capacity are caused by erroneous tank parameter settings or erroneous sensor parameter in which both are caused by human faults. Erroneous filtering of fuel level value is caused by fault in low pass filter equation. Erroneous mapping of voltage to volume in percentage is caused by fault in mapping look up table. Whereas erroneous fuel level sensor value is caused by either electrical or mechanical fault in fuel sensor. Figure 5.15, shows the fault tree for this deviation.

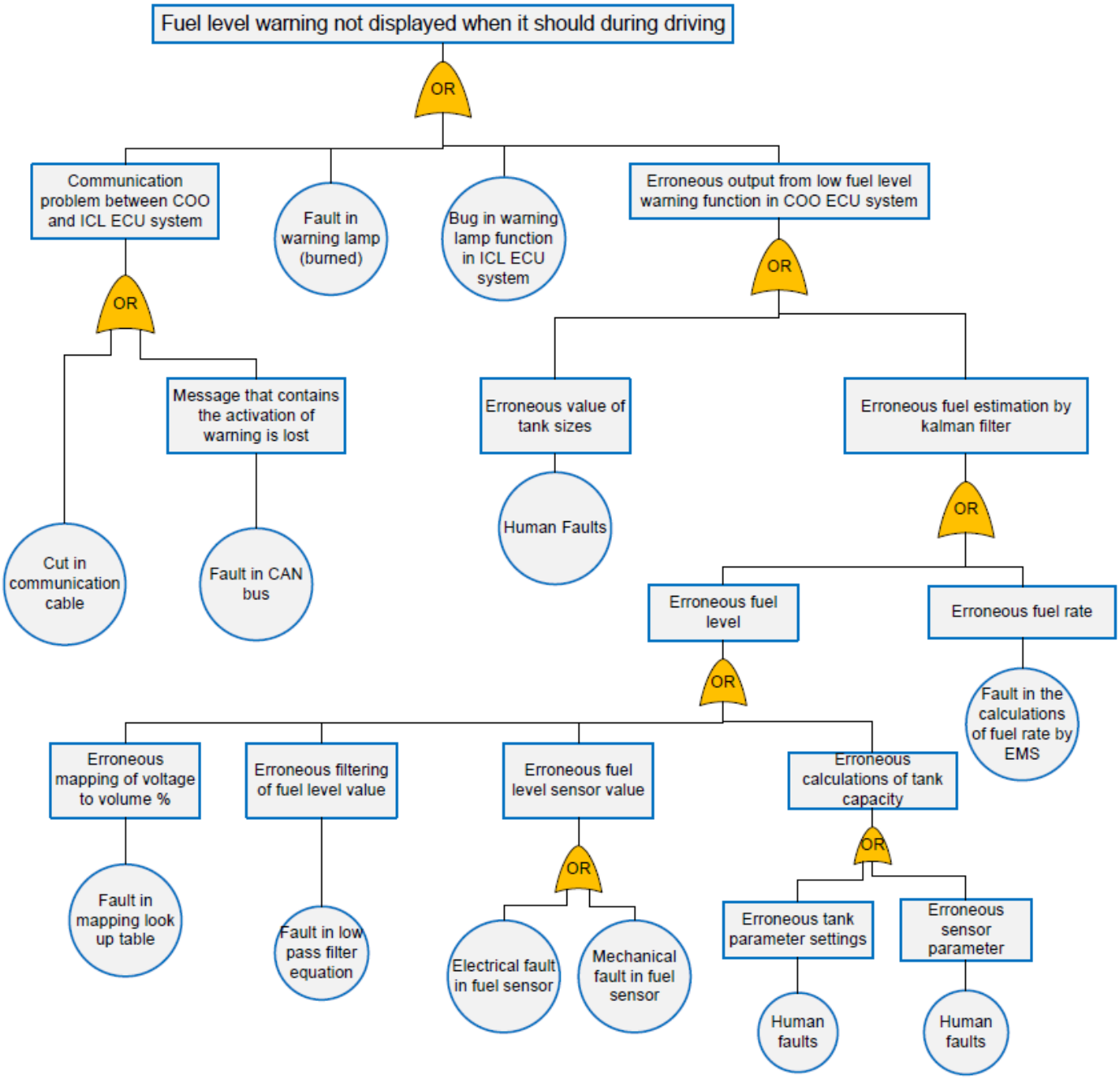


Figure 5.15: Fault tree for the deviation "Fuel level warning not displayed when it should"

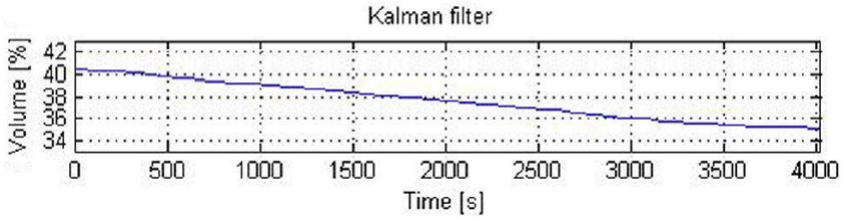


Figure 5.16: Simulation result for the system when KF driven only by fuel consumption [23]

A verification of the system design has been found for fuel level display system. So based on what have been found for the system, the next requirement in the product development at the system level that has been followed is requirement 7.4.8.1 verification of the system design. This requirement states that

- 7.4.8.1 *The system design shall be examined for compliance and perfection with respect to the technical safety concept using the verification methods as listed in Table 2.11 in the background subsection 2.4.3.*

Three methods have been used for the verification of the system design and they correspond to three methods from Table 2.11: System Design Walk-through, Simulation and System Prototyping and Vehicle Tests. System Design Walkthrough has been conducted by the author of this master thesis since there was no verification for a number of requirements through simulation and system prototyping methods because these requirements were not easy to verify. A number of AERs that correspond to FSRs and TSRs have been verified using these three methods. Other allocation element requirements are verified during the software verification part. Thus software verification is complementary to system verification where each verification level (system or software level) takes care of a number of requirements. AERs that have been verified using System Design Walkthrough, simulation and systems prototyping and vehicle tests are AER_201_4, AER_201_09, AER_201_11, AER_201_20, AER_201_21 and AER_201_42. Specifically, AERs that have been verified using System Design Walkthrough method are AER_201_4 and AER_201_21. The verification using simulation and system prototyping methods is done by Peter Wallbeck [23]. The results of simulation and system prototyping are presented in the following figures. The first sets of figures are the results from simulation. Figure 5.16 shows that KF algorithm can still estimate the fuel level even when there's no input from fuel level sensor.

Figure 5.17 shows that KF algorithm can estimate the fuel level even when theres no input about fuel consumption. Thus, this simulation result verifies AER_201_20 and AER_201_42.

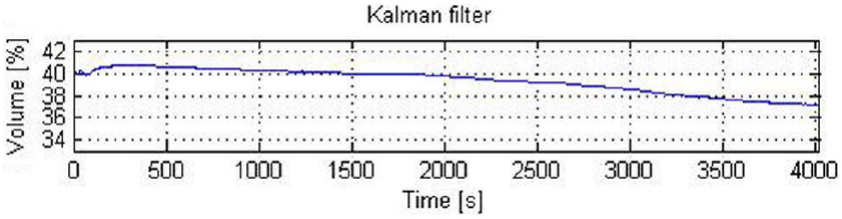


Figure 5.17: Simulation result for the system when KF driven only by fuel level sensor [23]

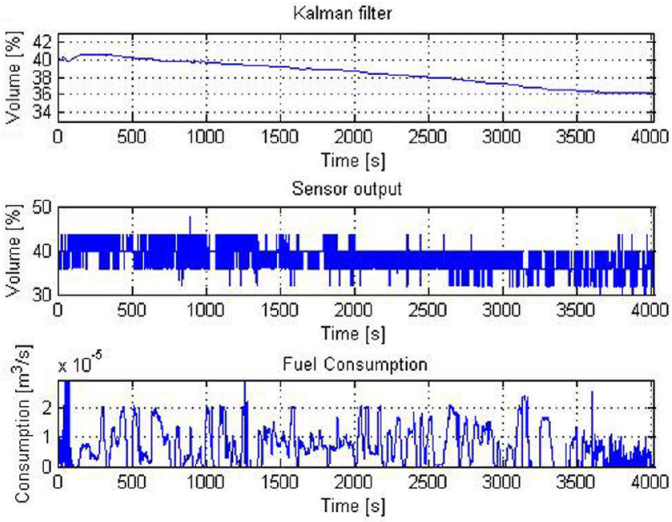


Figure 5.18: Simulation result for the system when KF driven by fuel consumption and fuel level sensor [23]

Figure 5.18 shows that KF estimate the fuel level by depending on two inputs which are: fuel level sensor and fuel consumption. This simulation result verifies AER_201_11. Moreover, in order for KF to estimate fuel level, a pre-filtering using low pass filter is implemented. Thus this simulation result verifies AER_201_9.

Using the verification method system prototyping and vehicle test, the system gave the same results as the simulation results but with lower resolution. Thus, it means that the system works properly when implemented in a Scania vehicle. Figure 5.19 shows KF when it's driven only with fuel consumption on a real vehicle. Other scenarios such as KF driven by sensor or KF driven by both of the input signals (fuel level sensor and fuel consumption) gives the same result as the simulation results but those results are not presented here as they are the same as the simulation results. For more information about the simulation, please refer to Peter Wallbeck thesis report [23].

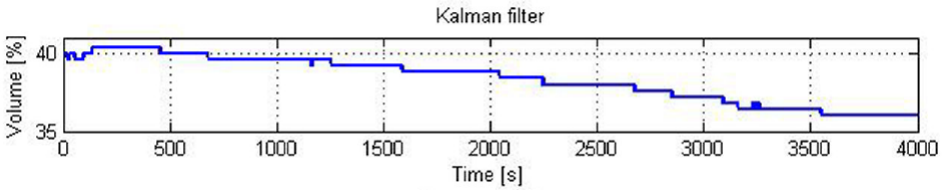


Figure 5.19: Vehicle test results for the system when KF is driven only by fuel consumption [23]

5.5 Software Architectural Design (Pt 6, Cl 7)

The objective of the software architectural design is to develop the software architecture as well as to verify it. The software architectural design represents all the software components along with the interactions, data paths, and interfaces among these components. The software architectural design should be developed in compliance with the software safety requirements. However, the system is already under production and no there are no software safety requirements have been found in the requirement documents of fuel level display system. Since the system design presented in section 5.4 contains all the software components, interaction and interface among those components. Thus the system design is also considered as the software architectural design. A verification of the software arch design has been found for fuel level display system. So based on what have been found for the system, the next requirement in the product development at the software level that has been followed is requirement 7.4.18 verification of the software arch design. This requirement states that

- 7.4.18 *the software architectural design shall be verified in accordance and by using the software architectural design verification methods listed in Table 2.12 [17] in the background subsection 2.4.3.*

The method that has been used for the verification of the software architecture design is formal verification. Model checking approach has been used for the verification of the software architectural design as presented in Ali and Muhammad thesis work [30]. In their work the software architecture design has been verified against these requirements: AER_201_11, AER_201_12, AER_201_13, AER_201_14, AER_201_15, AER_202_2, AER_202_3, and AER_202_5.
However, the requirements that are safety related are AER_201_11 and AER_202_2. Thus, these requirements are in significance for this thesis. Figure 5.20 shows the verification model for the requirements mentioned above using Model checking approach.

Table 5.9 shows the safety requirements that have been verified during the *system design* and *software architectural design* as well as the verification

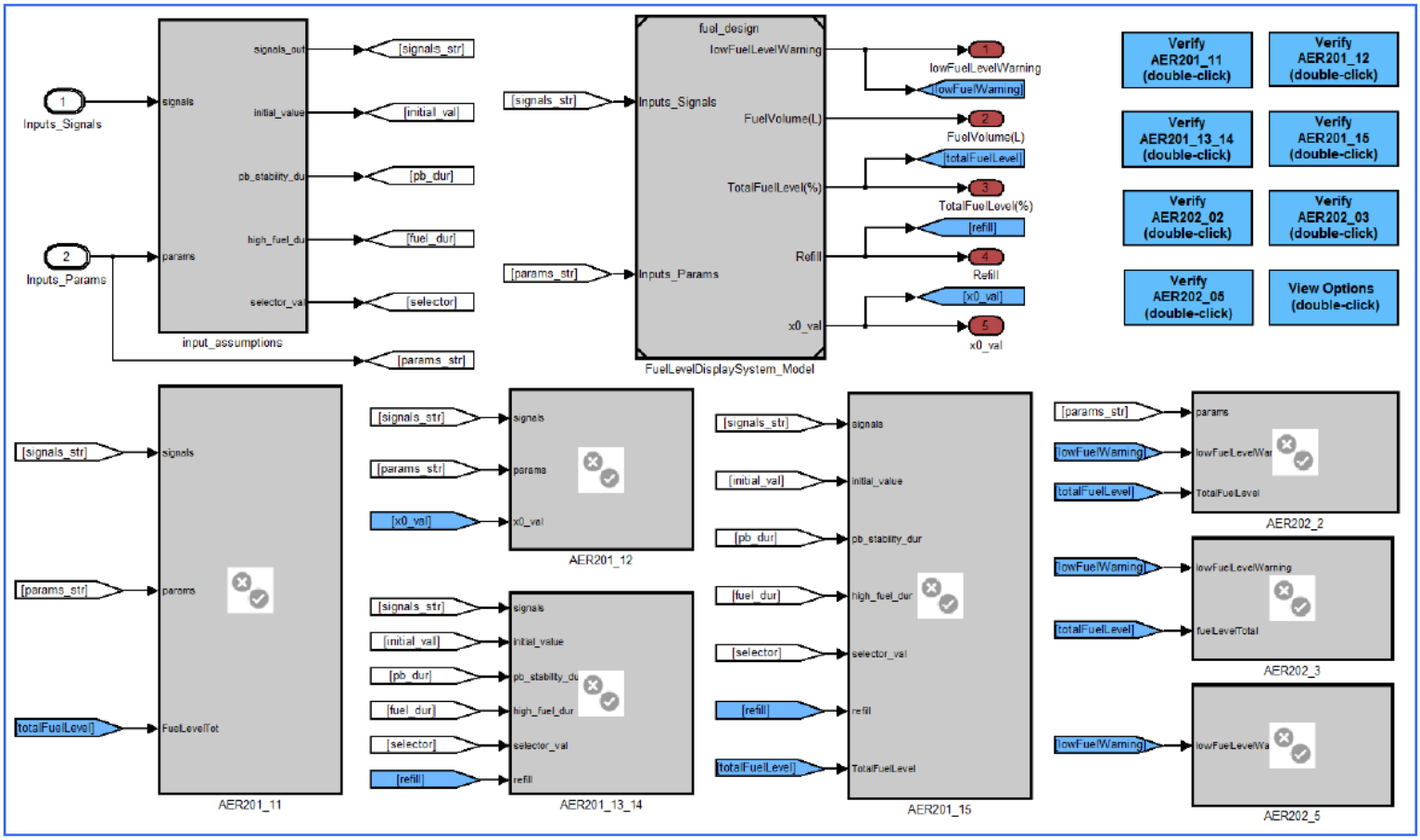


Figure 5.20: The verification model for the requirements of FLEDS using Model checking [30]

Table 5.9: Safety related requirements of FLEDS that have been verified along with the verification methods used

Verified AER	Corresponding FSR ID	Corresponding TSR ID	Verification method	Phase
AER_201_9	FSR 1.1	TSR 1	• Simulation • System prototyping and vehicle tests	During system design
AER_201_11	FSR 2	TSR 5	• Simulation • system prototyping and vehicle tests • Formal verification using model checking	During system design • During software architectural design
AER_201_20	FSR 1.2 & FSR 1.4	TSR 2 & TSR 4	• Simulation S • System prototyping and vehicle tests	During system design
AER_201_21	FSR 1.3	TSR 3	System design walkthrough	During system design
AER_201_42	FSR 1.4	TSR 4	• Simulation • System prototyping and vehicle tests	During System design
AER_202_2	FSR 3.1	TSR 6	Formal verification using model checking	During software architectural design
AER_202_4	FSR 3.2	TSR 7	System design walkthrough	During System design

method used for verification.

5.6 Item Integration and Testing (Pt 4, Cl 8)

In item integration and testing, the integration and testing is done at three levels: Integration and testing of the hardware and the software of each element that compose an item, integration and testing of all the elements that compose an item and the integration and testing of the item with other items within a vehicle. The objectives of item integration and testing are: to test the compliance with safety requirements as well as to verify that the system design implements the safety requirements correctly. Item integration and testing is carried out through the following phases:

- Hardware and software integration and testing
- System integration and testing
- Vehicle integration and testing

In order to be able to see which testing level at Scania corresponds to which level in ISO26262, we need to have a look at the testing levels implemented for fuel level display system at Scania. The testing levels that have been implemented for the system are as follows:

- ECU system test
  The ECU system test integrates the hardware and software of an ECU in order to verify the inputs, outputs, performance and robustness.
- Complete vehicle system test and complete vehicle integration test
  These tests are performed to test the system of interest on the vehicle as well as testing the system with other systems within the vehicle. It aims at verifying the communication on the CAN bus as well as the basic functionality of the user functions.

Now, a mapping of the testing levels at Scania to ISO 26262 is presented below:

- Hardware-Software integration and testing
  The test level at Scania that best matches the hardware and software integration and testing is ECU system test as the ECU system test integrates the hardware and software of an ECU system.
- System integration and testing
  The test level at Scania that best corresponds to system integration and testing is lab integration test.
- Vehicle integration and testing
  The test level at Scania that best corresponds to vehicle integration and testing is complete vehicle system test and complete vehicle integration test. At Scania, this level is conducted in vehicles. Table 5.10 shows the mapping of Scania to ISO 26262

Table 5.10: Mapping of Scanias testing levels for FLEDS to item integration and testing in ISO 26262

ISO 26262	Corresponding testing levels for fuel level display system at Scania
Hardware software integration and testing	ECU system test
System integration and testing	Lab Integration test
Vehicle integration and testing	Complete vehicle integration test and complete vehicle system test

After checking the requirements of item integration and testing, requirement 8.4.2.2.4 of the standard is found to be implemented in FLEDS at Scania. The requirement is as follows:

- 8.4.2.2.4 *The consistent and correct implementation of the external and internal interfaces at the hardware-software level shall be demonstrated using feasible methods [17] given in Table 5.11*

Table 5.11: Consistent and correct implementation of external and internal interface at the hardware-software level [17]

Methods		ASIL			
		A	B	C	D
1a	Test of external interfaces[a]	+	++	++	++
1b	Test of internal interfaces[a]	+	++	++	++
1c	Interface consistency check[a]	+	++	++	++

[a] Interface tests of the test object include tests of analogue and digital inputs and outputs, boundary tests and equivalence-class tests to completely test the specified interfaces, compatibility, timings and other specified ratings for the test object. Internal interfaces of an ECU can be tested by static tests for the compatibility of software and hardware as well as dynamic tests of Serial Peripheral Interface- (SPI) or Integrated Circuit- (IC) communications or any other interface between elements of an ECU.

In order to be able to know which testing method of hardware and software integration and testing Scania have used for FLEDS, a brief explanation about internal and external interfaces is required. Thus, an interface means the interaction among components (hardware/software). External interfaces presents the interaction (i.e. input and output) of the ECU system of interest with other ECUs through the CAN bus. Since ECU system test tests against the communication on the CAN and the inputs and outputs passed among ECUs. Thus, as a conclusion, ECU system test for fuel level display system uses +emphtesting of external interfaces method. See Figure 5.21 for the original sample for one of the test cases regarding testing with respect to communication and the inputs and outputs that is carried out in ECU system test in Scania. Internal interfaces mean an interaction among the allocation elements of the ECU system of interest, as well as the interaction between allocation element and sensors and actuators. ECU system test

4 **TestCase: Tc_AE201_13 - Passed** 2012-08-15T23:25:30.085 - 2012-08-15T23:28:50.582

Testcase description:
Scriptfile: \\n27466\ta\llab2\TC_UF\Tc_AE201_13.py
Testcase overall results: http://n28268:8080/ReportServer_SQLEXPRESS?/Rapport/resultatrapport&filter=1&filter=2&filter=3&filter=6

Tests

Timestamp	Validation	Description	Outcome
23:26:29.306	Act1 (Test step 10002.4.1.3): Unknown2	DashDisplay:K:FuelLevel is expected to be == 0.0 on YELLOW_1 bus. //Expression: '0.0==0.0' ()	Passed
23:26:29.391	Act1 (Test step 10002.4.1.3): Unknown2	FuelEconomy:E:FuelRate is expected to be == 30.0 on RED_1 bus. //Expression: '30.0==30.0' ()	Passed
23:26:50.211	Act2 (Test step 10002.4.2.3): Unknown2	DashDisplay:K:FuelLevel is expected to be == 0.0 on YELLOW_1 bus. //Expression: '0.0==0.0' ()	Passed
23:27:11.320	Act3 (Test step 10002.4.3.3): Unknown2	DashDisplay:K:FuelLevel is expected to be >< [21.0, 28.0] on YELLOW_1 bus. //Expression: '21.0 <= 21.6 <= 28.0' ()	Passed
23:27:31.861	Act4 (Test step 10002.4.4.3): Unknown2	DashDisplay:K:FuelLevel is expected to be >< [48.0, 58.0] on YELLOW_1 bus. //Expression: '48.0 <= 48.8 <= 58.0' ()	Passed
23:27:52.685	Act5 (Test step 10002.4.5.3): Unknown2	DashDisplay:K:FuelLevel is expected to be >< [70.0, 82.0] on YELLOW_1 bus. //Expression: '70.0 <= 76.0 <= 82.0' ()	Passed
23:28:13.509	Act6 (Test step 10002.4.6.3): Unknown2	DashDisplay:K:FuelLevel is expected to be >< [93.0, 100.0] on YELLOW_1 bus. //Expression: '93.0 <= 93.6 <= 100.0' ()	Passed

Figure 5.21: Original sample for one of the test cases regarding ECU system test done for FLEDS at Scania

aims at verifying AERs that are allocated to the ECU system of interest and since those requirements specifies the interaction among AEs and sensors and actuators. Thus, ECU system test tests internal interfaces as well. Therefore, as a conclusion, ECU system test uses *test of internal interfaces* method. Table 5.12 shows the corresponding test cases and results for the requirements covered.

In Figure 5.21, the description column shows that the testing covers the inputs, the outputs as well as the communication over CAN. So for example you can see in Act1 what should be expected on the yellow bus and the output value as well.

After checking the requirements of vehicle integration and testing, requirement 8.4.4.2.4 of the standard is found to be implemented in fuel level display system at Scania. The requirement is as follows:

- 8.4.4.2.4 *The consistency and correctness of the implementation of the external interfaces at the vehicle level shall be demonstrated using feasible test methods as given in Table 5.13 [17].*

Table 5.12: Test cases and results for the ECU system test in Scania that corresponds to hardware software integration and testing

Hardware software integration and testing					
AER	Corresponding FSR	Corresponding TSR	Test case	Results	Testing Method
AER_201_2	Not a safety related requirement	Not a safety related requirement	TC_AE201_2	Passed	Testing of internal and external interfaces
AER_201_6	Not a safety related requirement	Not a safety related requirement	TC_AE201_6		
AER_201_9	FSR 1.1	TSR 1	TC_AE201_9		
AER_201_11	FSR 2	TSR 5	TC_AE201_11		
AER_201_12	Not a safety related requirement	Not a safety related requirement	TC_AE201_12		
AER_201_13	Not a safety related requirement	Not a safety related requirement	TC_AE201_13		
AER_201_14	Not a safety related requirement	Not a safety related requirement	TC_AE201_14		
AER_201_15	Not a safety related requirement	Not a safety related requirement	TC_AE201_15		

Table 5.13: Consistent and correct implementation of external and internal interface at the vehicle level [17]

Methods		ASIL			
		A	B	C	D
1a	Test of external interfaces[a]	o	+	++	++
1b	Test of interaction/communication[b]	o	+	++	++

[a] An interface test at the vehicle level tests the interfaces of the vehicle systems for compatibility. This can be done statically by validating value ranges, ratings or geometries as well as dynamically during operation of the whole vehicle.

[b] A communication and interaction test includes tests of the communication between the systems of the vehicle during runtime against functional and non-functional requirements.

Test of interaction/communication aims at testing the communication among the ECUs of the vehicle during the runtime. Complete vehicle system test, complete vehicle integration test and lab integration test test the communication on the CAN bus during the basic functionality of different ECU systems. The testing is conducted in integration lab (ILab) and sometimes even in lab vehicles. As a conclusion, complete vehicle system test, complete vehicle integration test and lab integration testing uses *test of interaction/communication* method during testing.

Table 5.14: Test cases and results for the vehicle integration system test and lab integration test for FLEDS in Scania that corresponds to vehicle integration and testing and system integration and testing in ISO26262

Vehicle integration and testing				
Test case	Description	Testing Environment	Result	Testing method
TC 18.1	This test case has a number of scenarios regarding the basic functionality of fuel estimation and the communication scenario that is expected to happen when estimating and displaying the fuel level on the fuel gauge.	• ILab • Vehicle	Passed	Testing of interaction/ communication
TC 18.5	This test case has a number of scenarios regarding the basic functionality of low fuel level warning and the communication scenario that is expected to happen when displaying warning on the lamp.			

6. SAFETY CASE OF FLEDS

The purpose of this chapter is to build the safety case of FLEDS. The case is built by using the theory of building a safety case in chapter 2 (more precisely section 2.5) and by using the evidence that has been collected and or provided in chapter 5. The evdience that has been collected and or provided in is used to create the process and product-based arguments. These arguments are combined to form the safety case of FLEDS. D-Case editor tool has been used for the creation of the safety case because of its user friendly GUI and its support for modularity and GSN as mentioned before in section 2.9. More details about how the evidence from chapter 5 has been used for the creation of the safety case is presented further in this chapter. Functional breakdown pattern in Figure 2.18 has been used in the creation of FLEDS safety case. We have already collected and provided evidence about the safety of the system functions of FLEDS and the interactions among these functions. Therefore, this pattern has been reused in the creation of the safety case. Functional breakdown pattern has been used in the argument given in figure Figure 6.2.

Figure 6.1, shows the goal structure for FLEDS. In the structure, a clear separation between process and product-based arguments has been maintained. Moreover, the top level goal is broken down by strategy S1: Argument over the behaviour of the system and the process followed during the development life-cycle. We proceed with the safety case by arguing about the behaviour of the system (product-based argument).

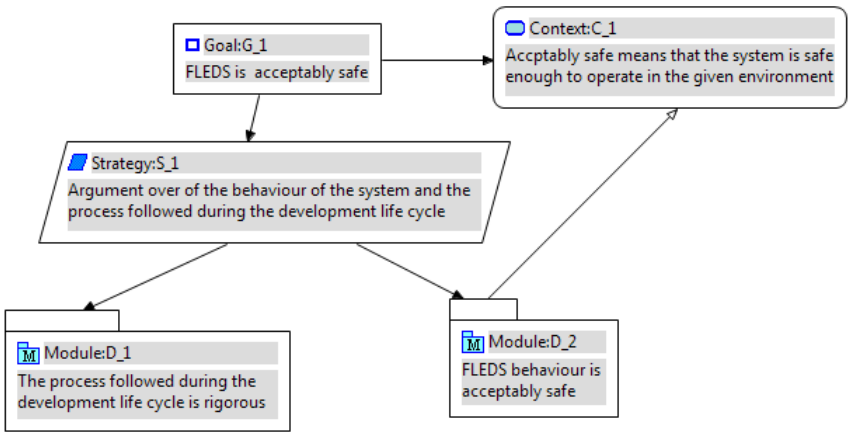


Figure 6.1: Goal structure of FLEDS

Figure 6.2 shows the argument for module D_2: FLEDS behaviour is acceptably safe. This goal structure reuses the functional breakdown pattern. The goal structure provides an argument to show that the top level goal G_2 : FLEDS behaviour is acceptably safe, is supported by evidence regarding the safety related functions, the interactions among these functions and the hardware of FLEDS. Modules D_3 (Safety related functions of FLEDS are acceptably safe), D_4 (Interaction between system functions are non hazardous) and D_5 (Hardware related to FLEDS is acceptably safe) have been used to provide arguments for goal G_2.

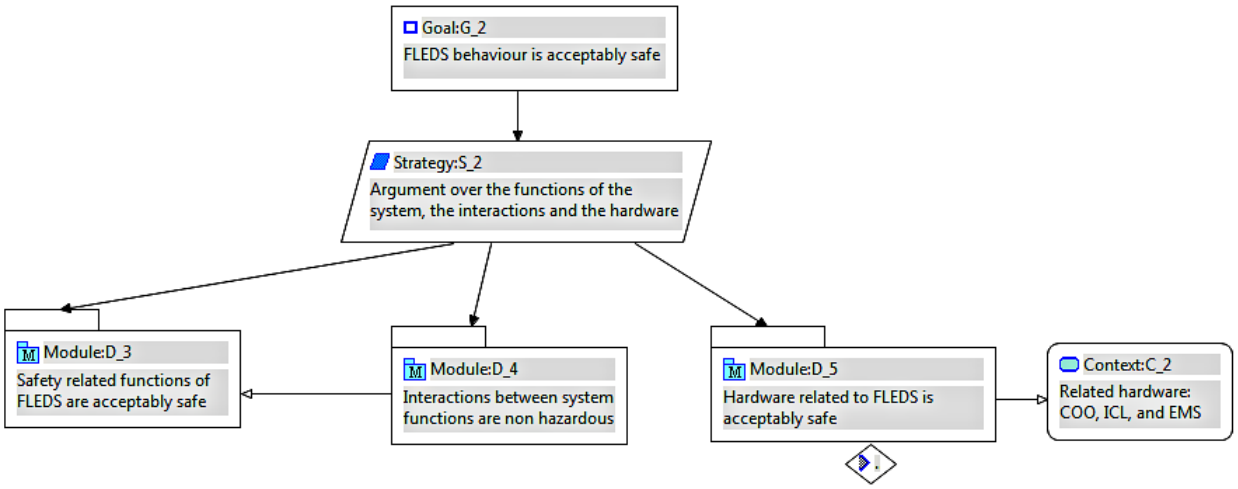


Figure 6.2: Goal structure for the product-based argument

The goal structure in Figure 6.3 provides an argument that supports the top level goal G_3: Safety related functions of FLEDS are acceptably safe.

This goal is supported by an argument over the safety of fuel level estimation function (G_4) and fuel level warning function (G_5) by using strategy S_3: Argument over the safety of safety related functions. Thereafter, goal G_4 is supported by arguments over the robustness of fuel level estimation algorithm (Module D_6), hazard identification and analysis (Module D_7) and mitigation/elimination of the hazards associated to fuel level estimation function (Module D_8). Likewise, goal G_5 is supported by arguments over hazard identification and analysis (Module D_7) and mitigation/elimination of the hazards associated to fuel level warning function (Module D_9). Goal structures for Modules D_6, D_7, D_8 and D_9 are presented further in the next pages.

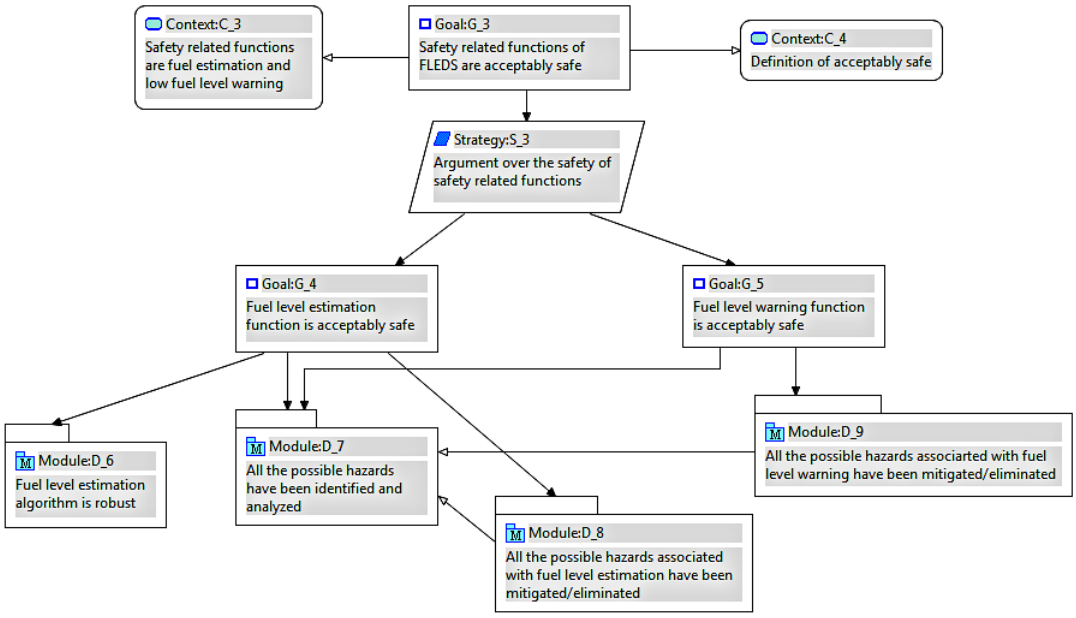


Figure 6.3: Goal structure for the Module D_3

The goal structure in Figure 6.4 provides the argument to show that the top level goal G_5 is supported by evidence E_1: Standard deviation analysis and E_2: Simulation results. This goal structure merges the findings in chapter 5 concerning the software architectural design in section 5.5, more precisely simulation results (Figure 5.16, Figure 5.17 and Figure 5.18), system prototyping and vehicle test in Figure 5.19 and standard deviation analysis in appendix A. Evidence E_1 is thus used to support goal G_7: Kalman filter gives a steadier estimate with less deviation whereas evidence E_2 is used to support goals G_8: Filter's output is rarely shows a false increase in the fuel level and G_9: Kalman filter is not easily affected by fuel movements in the tank.

The goal structure in Figure 6.5 provides the argument to show that

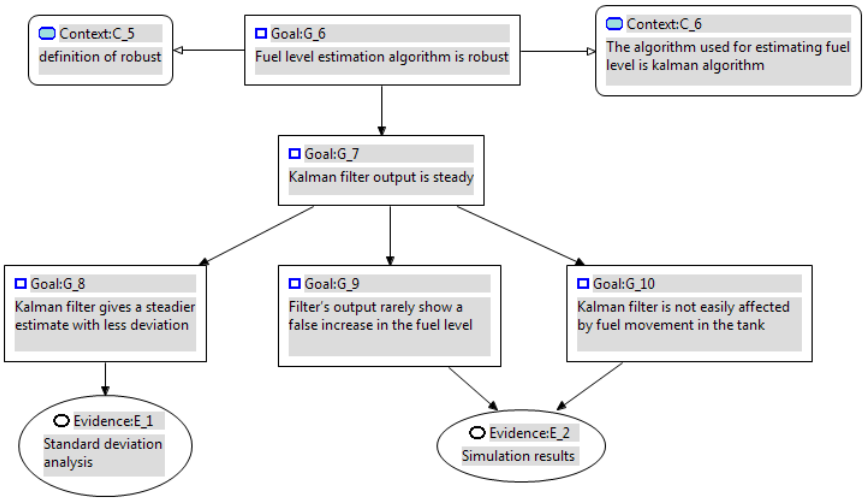


Figure 6.4: Goal structure for the Module D 6

the top level goal G_6 is supported by evidence E_3: Adapted HAZOP (Table 5.1), E_4: FTA figures (Figure 5.11 to Figure 5.15) and E_5: FMEA tables (Table 5.6 to Table 5.8). This goal structure merges the findings in chapter 5 concerning Hazard analysis in section 5.1 and system design in section 5.4. Evidence E_3 is thus used to support goal G_7: All the possible hazards have been identified, G_8: All the possible hazards have been analyzed to their possible causes and G_9: All the possible hazards have been analyzed to their possible consequences, then evidence E_4 is used to support goal G_8 and E_5 is used to support goal G_9.

Figure 6.6 shows the argument for module D_8: All the possible hazards associated with fuel level estimation have been mitigated/eliminated. The goal structure provides an argument to show that the top level goal G_10 is supported by evidence regarding the mitigation of hazard H1, H2 and H3. Modules D_10 and D_11 have been used to provide arguments regarding mitigation these hazards.

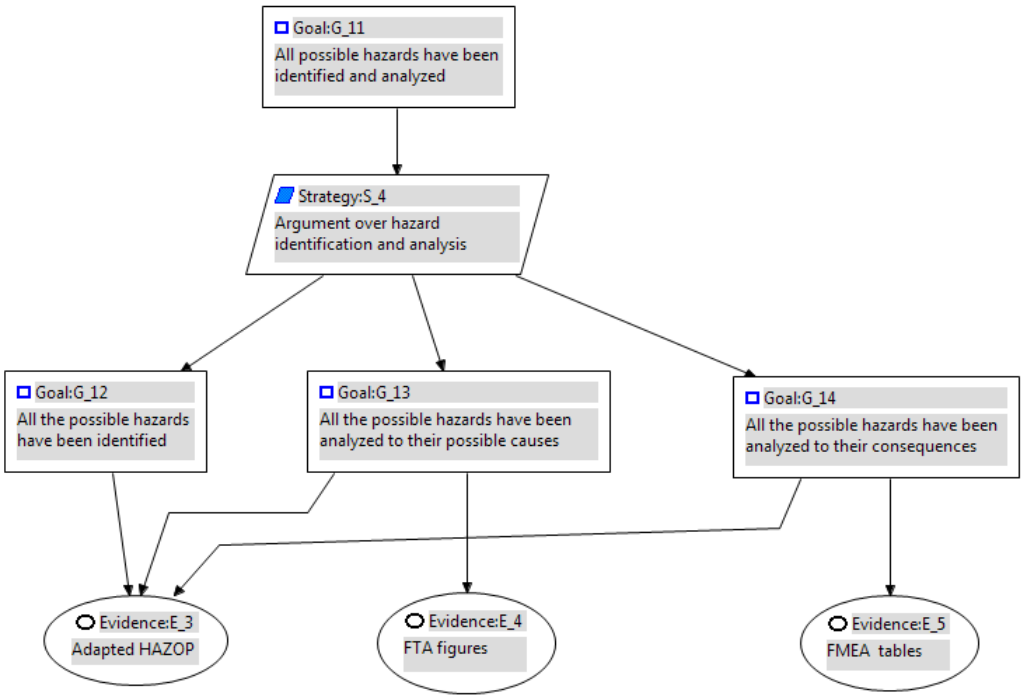


Figure 6.5: Goal structure for the Module D 7

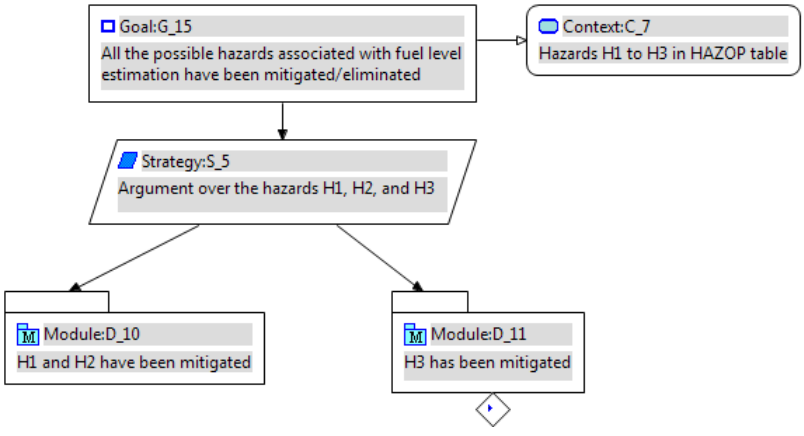


Figure 6.6: Goal structure for the Module D 8

Module D_11: H3 has been mitigated, is not developeds further since that hazard is related to the hardware and we haven't cover the hardware in this thesis due to time limitation. Module D_10: H1 and H2 have been mitigated, is supported by the goal structure that is shown in Figure 6.7. This goal structure merges the findings in chapter 5 concerning the verification of the system and the software design(Figure 5.6, Figure 5.7 and Table 5.9) as well as the findings concerning the identification of the SGs, FSRs and TSRs (Table 5.2, Table 5.3 and Table 5.4 respectively).

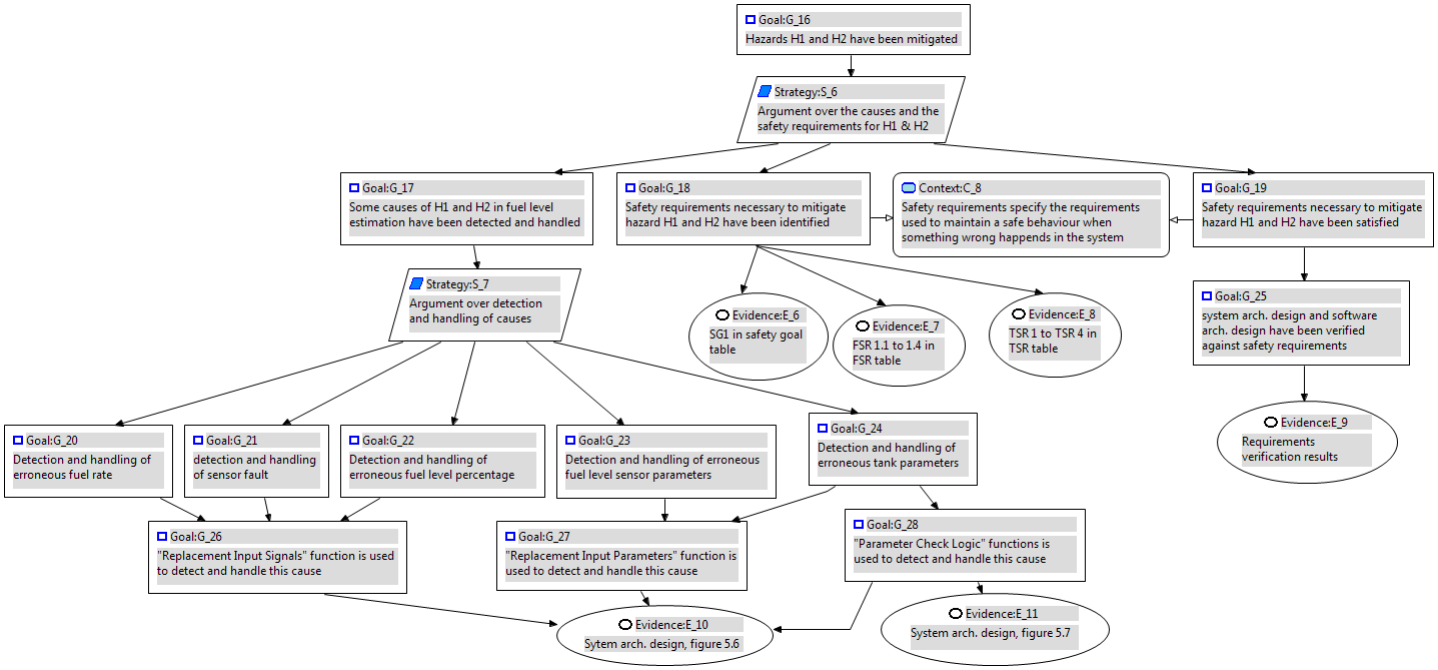


Figure 6.7: Goal structure for the Module D 10

Figure 6.8 shows the argument for module D_9: All the possible hazards associated with fuel level warning have been mitigated/eliminated. The goal structure provides an argument to show that the top level goal G_24 is supported by two modules D_12 and D_13. Module D_12 is used to provide an argument that hazard H4 is negligible whereas module D_13 is used to provide an argument that hazard H5 has been mitigated.

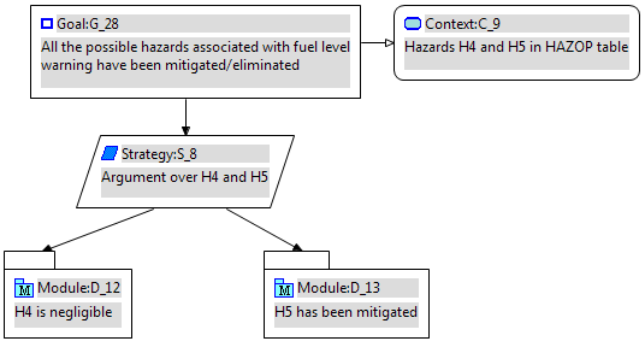


Figure 6.8: Goal structure for the Module D 9

Figure 6.9 shows the goal structure for module D_12: H4 is Negligible. This goal structure merges the findings in chapter 5 concerning hazard analysis and risk assessment, more precisely ASIL classification in Table 5.1.

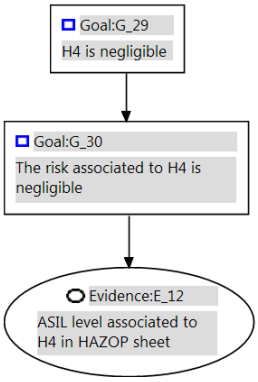


Figure 6.9: Goal structure for the Module D 12

Figure 6.10, shows the goal structure for Module D_13: Hazard H5 has been mitigated. This goal structure merges the findings in chapter 5 concerning the system design and the verification of the system and the software design (Figure 5.10 and Table 5.9). Moreover, this structure merges the findings concerning the identification of the SGs, FSRs and TSRs (Table 5.2, Table 5.3 and Table 5.4 respectively).

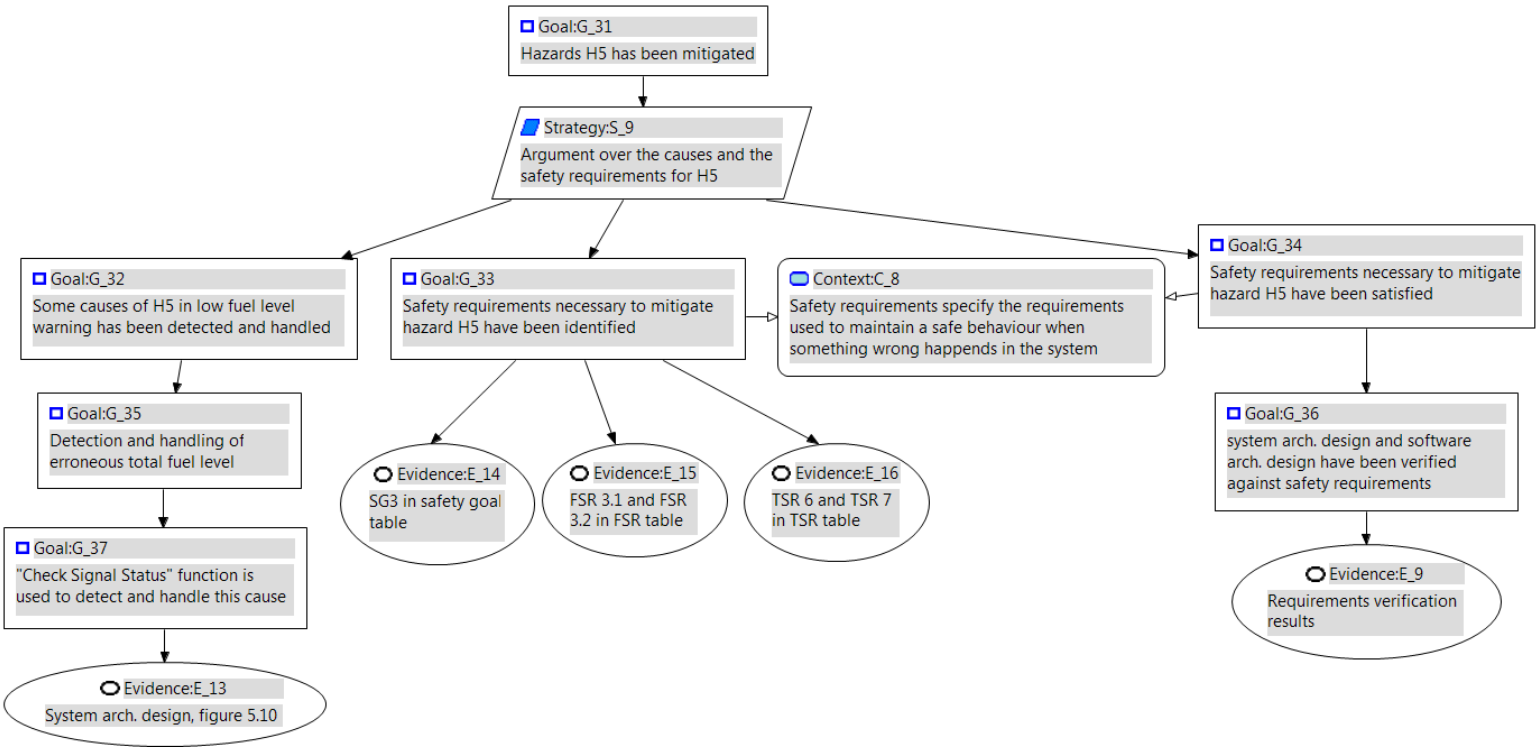


Figure 6.10: Goal structure for the Module D 13

Back to the product-based argument given in Figure 6.2, furthermore, we need to support module D_4 . The goal structure in Figure 6.11 provides the argument to support module D_4. This goal structure merges the findings in chapter 5 concerning item integration and testing in section 5.6 . The top level goal G_34 is supported by evidence E_17: HW-SW integration and testing (Table 5.12) and E_18: vehicle integration and testing (Table 5.14).

Furthermore, we need to support process-based argument given in Figure 6.1. Process-based argument is supported by four sub-modules D_14, D_15, D_16 and D_17. Figure 6.12 shows the sub-modules that support the process-based argument. The goal structures to support each of these nodules are presented in the next few pages.

Figure 6.13, shows the goal structure for Module D_14: Requirements definition process is trustworthy. This goal structure merges the findings in chapter 5 concerning the identification of the SGs, FSRs and TSRs (Ta-

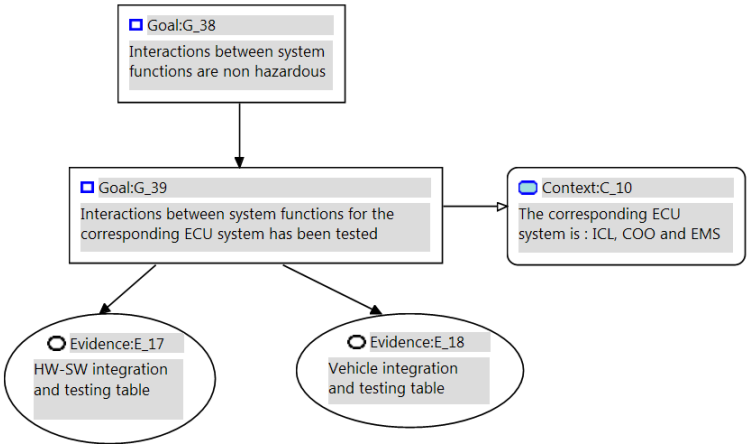


Figure 6.11: Goal structure for Module D 4

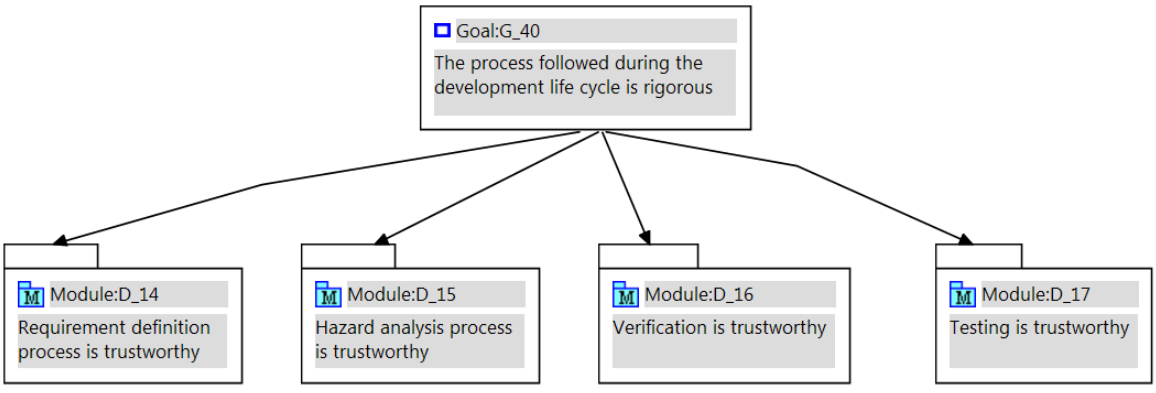


Figure 6.12: Goal structure for Module D 1

ble 5.2, Table 5.3 and Table 5.4 respectively).

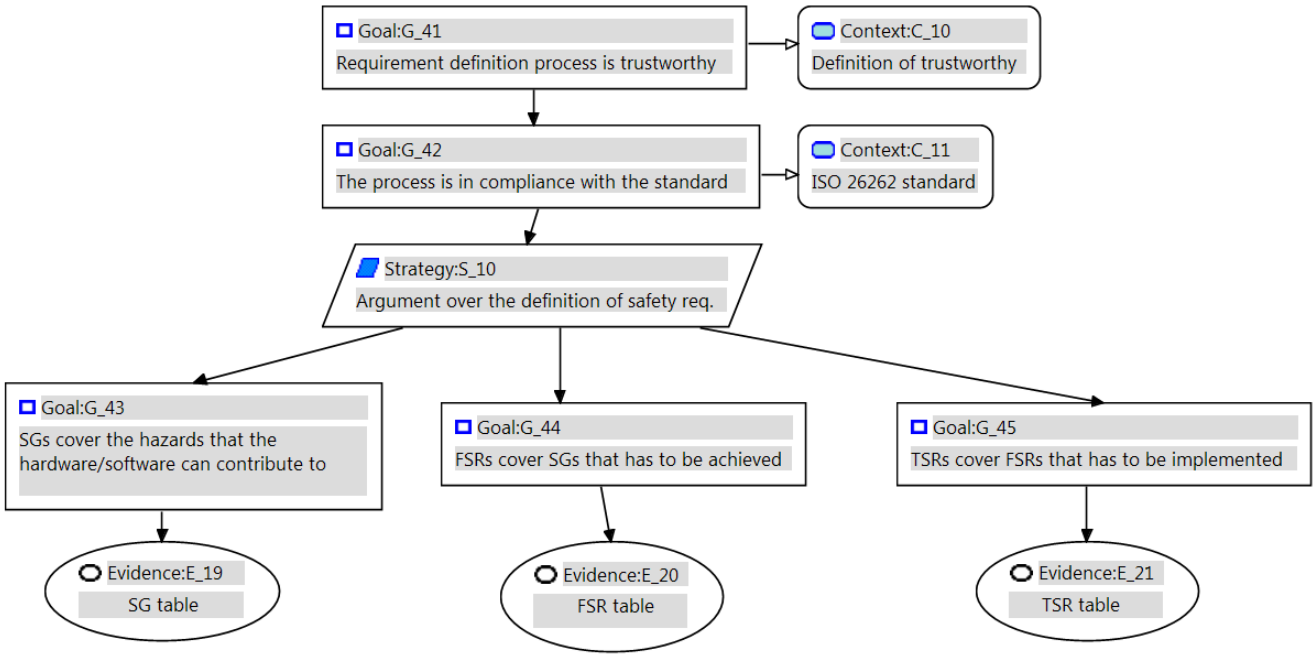


Figure 6.13: Goal structure for Module D 14

Figure 6.14 shows the goal structure for module D_15: Hazard analysis process is trustworthy. The goal structure merges the findings in chapter 5 concerning Hazard analysis in section 5.1 and system design in section 5.4. This goal structure reuses evidence E_3, E_4 and E_5 from module D_7 in Figure 6.5. The the top level goal G_42 is supported by evidence E_3: Adapted HAZOP (Table 5.1), E_4: FTA figures (Figure 5.11 to Figure 5.15), E_5: FMEA tables (Table 5.6 to Table 5.8) and E_22: Training and education.

Figure 6.15 shows the goal structure for module D_16: Verification is trustworthy. The goal structure merges the findings in chapter 5 concerning the verification of the system design in section 5.4 and the verifiaction of the software architectural design in section 5.5. This goal structure reuses evidence E_2 from module D_6 in Figure 6.4. The the top level goal G_47 is supported by evidence E_23: Verification model of the requirements (Figure 5.20), E_2: Simulation results (Figure 5.16, Figure 5.17 and Figure 5.18) and E_24: Verification results in Scania's vehicle (Figure 5.19).

Figure 6.16 shows the goal structure for module D_17: Testing is trustworthy. The goal structure merges the findings in chapter 5 concerning item integration and testing in section 5.6. This goal structure reuses evidence E_17 and E_18 from module D_4 in Figure 6.11. The the top level goal G_54 is supported by evidence E_25: Experience and education, E_26: Testing team is independent from design team, E_17: HW-SW integration and

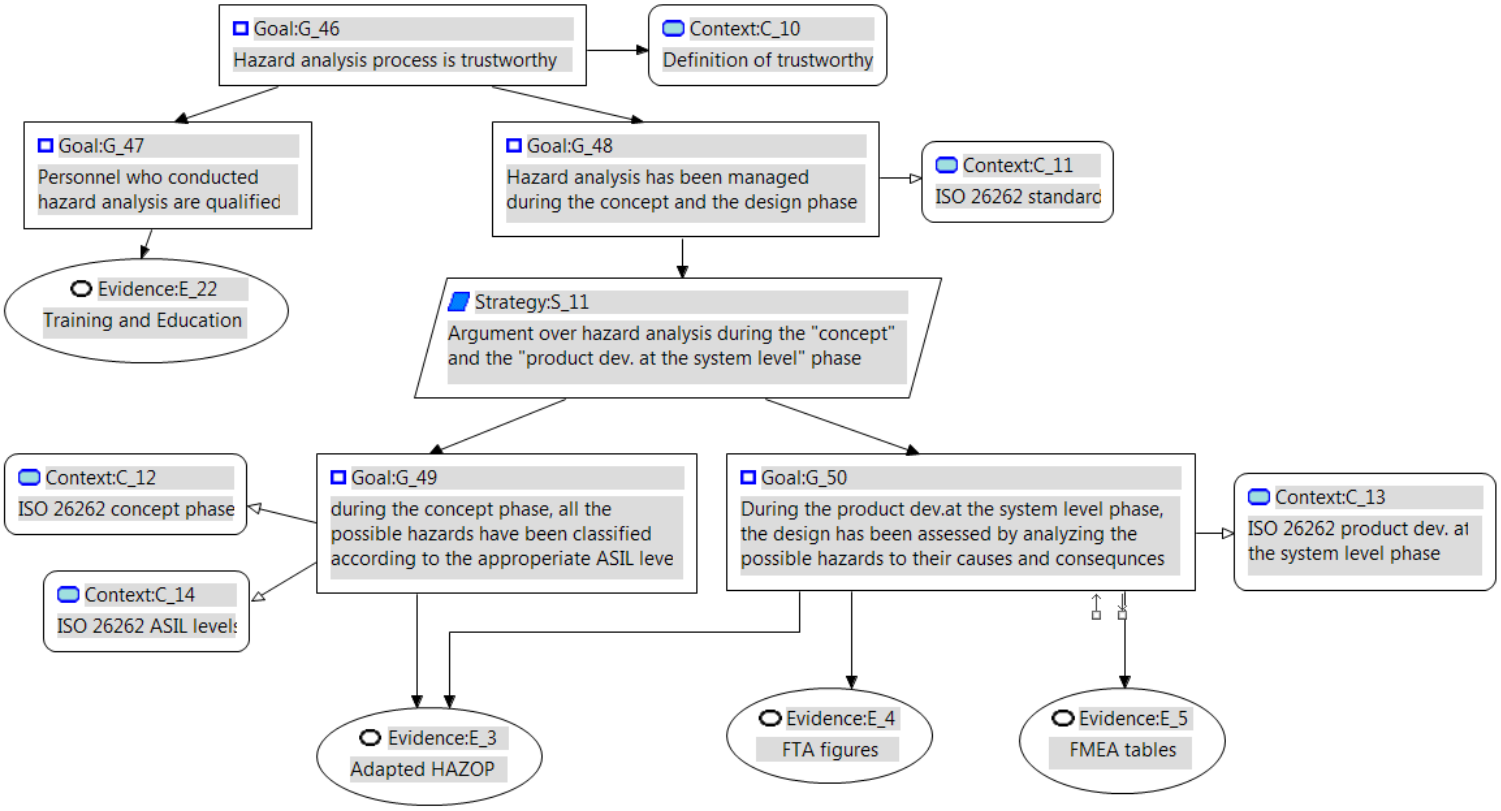


Figure 6.14: Goal structure for Module D 15

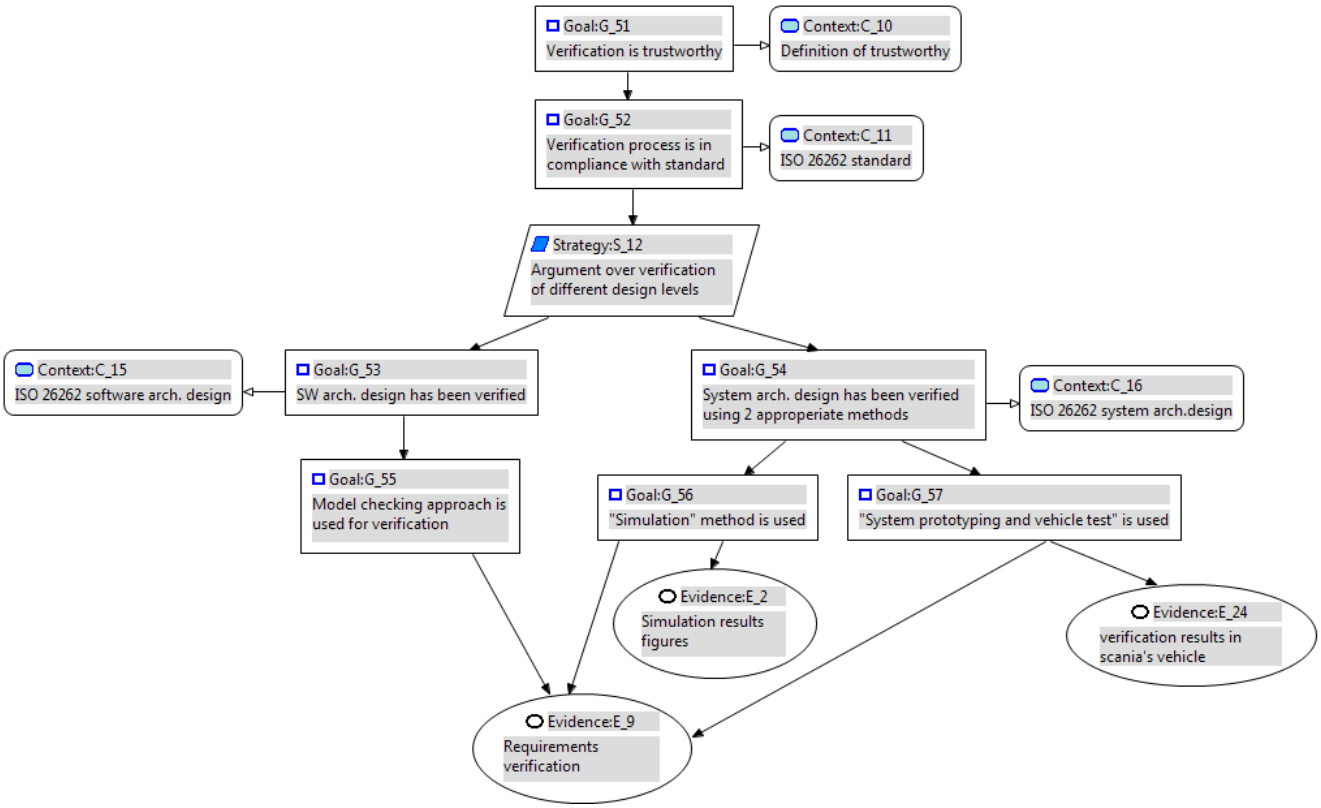


Figure 6.15: Goal structure for Module D 16

testing (Table 5.12) and E_18: Vehicle integration and testing (Table 5.14).

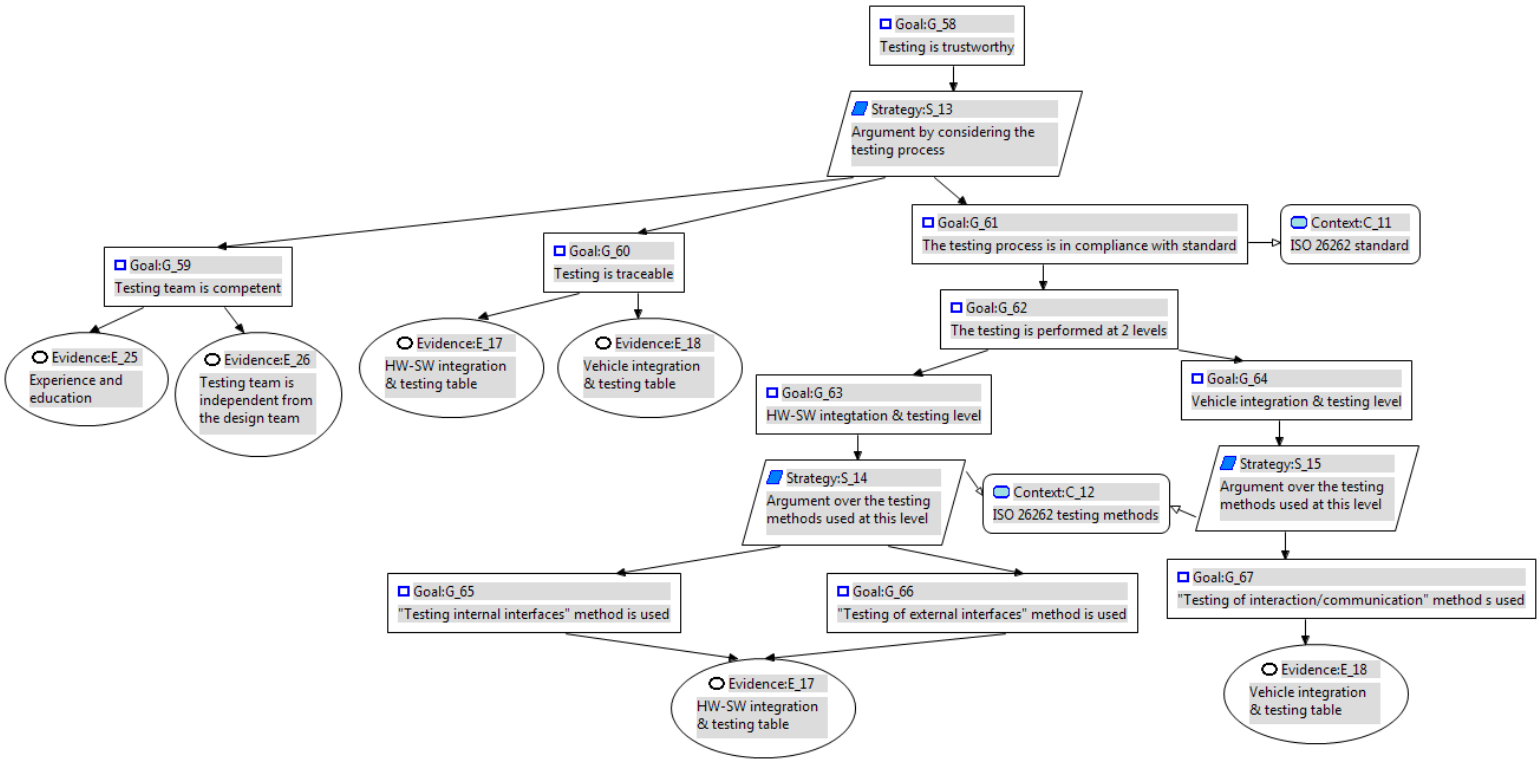


Figure 6.16: Goal structure for Module D 17

7. DISCUSSION

In this section, the author of this thesis present the experience concerning applying ISO26262 and building a safety case in a setting that is not in the scope of ISO26262. By comparing the life cycle presented in ISO26262 and the life cycle used in the company, the author of this thesis have realized that:

- The life cycle of the company doesn't consider some work-products that are compulsory by the standard. Such work-products are SGs, FSRs, and TSRs, as presented in chapter 5, more pecisely Table 5.2, Table 5.3, and Table 5.4.
- Traceability among life cycle work-products (i.e. SG, FSR, TSR, verification and testing) is quite essential and it must be seeked and retained. Presently, the company doesn't have any approach to maintain traceability in order to be in compliance with the standard. Valuable means to support traceability would be a model-based approach or tables with cross references as presented in chapter 5 when maintaining SGs, FSRs and TSRs for example.
- By deliberating ISO26262 with the company employees, the company has evolved curiousity in the standard.
- The practice of FTA in chapter 5 as recommended by ISO26262 has led to be advantageous to achieve a complemental analysis of the system. Therefore, the company has evolved curiousity in using the technique in the future.
- Building a safety case in compliance with ISO26262 is useful because it's easy to support evidence traceability since each clause followed will result in product and process-based evidence. As a result, the safety case more dependable because it's in compliance with the ISO26262.

By building the safety case in chapter 6, the author of this thesis has realized that:

- The product-based evidence should show that the system behaves in a safe way when a failure occurs. Moreover, the product-based evidence should emerge from activities such as verification (i.e. testing

and model checking) and simulation as presented in chapter 6. From the user point of view, the product behaviour is more essential than the process followed during the development life-cycle of the product. Therefore, at the first sight, product-based arguments may look like more important than the process-based arguments. However, there's a necessity for trust in the product-based evidence and therefore process-based arguments play an essential role too. Therefore, to have more persuasive safety case, the two types of the arguments should be supplied.

- Though we have applied ISO26262 and built the safety case for a simple system, the resulted safety case in chapter 6 is big and complex. As a result of this thesis, the author has realized that in case of complete conformity with ISO26262, even small systems will result in complex safety cases because the standard demands around 100 work-products that comes from fulfilling all the requirements from different phases of the standard's life-cycle. Therefore, managing the complexity of the safety case is laborious. Even though modularity and the use of patterns can be useful, however,safety case writers can write arguments with good structures only after having a long experience.

- If the company follows a common process for all the products that it develops then by having a clear separation between process and product-based arguments, the company can certify its process. By certifying its process, the company can focus more on the product-based argument rather than focusing on both of them. Moreover the company can generate process-based patterns that present evident comonalities. Therefore it's beneficial to have a clear separation between process and product-based arguments.

- It's not always clear how to supply evidence and the chance of baseless and fallacious safety cases is factual.

- The evidence can be unvisible either because it's undocumented or because the employees has no knowledge about it. Therefore, it's highly preferred to educate the staff with respect to ISO26262 in order to increase their knowledge about the necessary evidence required to build the safety case. By doing so, they will be able to provide quick and relevant information to safety experts in which will reduce the time of building the safety case.

The type of the safety case that is presented in chapter 6 is operational safety case because FLEDS is already under production. Therefore, the author of the thesis had to argue on the design of the system as it was given to author of this thesis. Therefore, there was no chance to affect the design

of the system. During building FLEDS safety case, a number of challenges and difficulties has been faced. The following challenges have been faced:

- It is hard to achieve a clear structure that contains evidence from various sources.
- In the beginning of the thesis, the author acted in a bias way since the company is a big one but later on the author has decided to act as an inspector and that made him get rid of the confirmation bias. So it's easy to fall in the bias trap.
- It's difficult to build a safety case without getting a chance to test the system and see how it really reacts in reality.
- It was hard to find relevant information related to the system because the system's developers have already left the company without documenting that much about the system. Moreover, Scania is not good at documentation. Relevant information about FLEDS has been gathered through interviews with employees.

8. CONCLUSION AND FUTURE WORK

In this chapter, the author of the thesis present what has been concluded throughout this theis regarding what is required in case Scania is interested in certifying its systems in compliance with ISO 26262. Moreover, future work for this thesis is presented further in this chapter.

8.1 Conclusion

In this report, we have presented a partial safety case that is in compliance with ISO26262 for one variant of FLEDS (one sensor with liquid fuel and one tank). Only few parts of the standard have been followed due to time limitation of this thesis. Discussion about how the evidence have been provided and or collected have been presented in this thesis as well as the use of the evidence to create product and process-based arguments. Moreover, this thesis doesn't provide a survey regarding how to verify the validity of the safety case since it's not in the scope of this thesis. However, one mechanism to check the validity of the safety case is by examining the safety case arguments with respect to the logical fallacies presented in the background section 2.7. Furthermore, in this thesis, a clear separation between product and process-based arguments have been maintained. Moreover, the lessons that have been learned during this thesis have been presented in chapter 7 in order to ease the adjustment of ISO26262. One of the important conlusions is that Scania needs to take care more about the documentation process in case it's interested in certification. Scania should document every single step during the development life-cycle because documentation should contain all the evidence about the process followed and the product behaviour. Moreover, Scania should have an approach for maintaining traceability because traceability is quite important in certification as it's used to check wheather the requirements have been implemented and verified against the design. Last but not least, it will take a lot of efforts for Scania to be completely in compliance with the standard because:

- The standard requires around 100 work products and these work products needs generation and documentation which will take time and

efforts.

- Scania's process will require a lot of change and adaptation to be in compliance with the standard. ASIL classification is an example for such an adaptation. Moreover, almost most of the systems are already in the production phase and thus it will take time and effort to make these systems in compliance with the standrd as it will require mapping of what these systems has to the standrd as well as providing the missing information required by the standard. More precisely, More work is required compared to what have been done in this thesis since in this thesis only some parts of the standard have been covered.

- It requires experience about ISO 26262 and this experience is not available recently as the standard was deployed in 2011. It takes time and efforts to gain such experience.

8.2 Future Work

In future, to achieve a complete safety case, we have to consider all the parts of ISO26262 as well as considering all the variants of the systems. Moreover, patterns can be generated from the partial safety case presented in this thesis in order to facilitate the argumentation. The goal is to concentrate on process-based arguments that are focusing on ISO26262 and other safety standards. Therefore, there's a necessity to survey the results in [2] in which a discussion about process line of safety arguments is presented.

Bibliography

[1] A. Johnsen, H. Kienle, K. Lundqvist and D. Sundmark, "Liability for Software in Safety-Critical Mechatronic Systems: An Industrial Questionnaire", *Proc. of the 2nd International Workshop on Software Engineering for Embedded Systems*, 2012.

[2] B. Gallina, O. Jaradat and I. Sljivo, "Towards a Safety- oriented Process Line for Enabling Reuse in Safety Critical Systems Development and Certification", *Post-proc. of the 35th IEEE Software Engineering Workshop (SEW-35), Heraclion, Crete (Greece)*, 2013.

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

[4] C.M. Holloway, "Safety Case Notations: Alternatives for the Non-Graphically Inclined?",In C.W. Johnson and P. Casely (eds.),*Proc. of the IET 3rd International Conference on System Safety, IET Press, Savoy Place, London*, 2008.

[5] DVA321 Safety Critical Systems Engineering Course, *Malardalen University*, http://www.mdh.se/studieinformation/VisaKursplan?kurskod=DVA321termin=20122sprak=en.

[6] F. Törner, P. Öhman, "Automotive Safety Case A Qualitative Case Study of Drivers, Usages, and Issues", *Proc. of the 11th IEEE High Assurance Systems Engineering Symposium*, 2008.

[7] G. Boman, "Modularization", Scania company presentation, 2010.

[8] G. Magnus, "Datakommunikation CAN", 2010.

[9] GSN Community Standard Version 1, November 2011, *http://www.goalstructuringnotation.info/GSN$_S$tandard.pdf*.

[10] H. Fujita, T. Hanawa, Y. Ishikawa, S. Kato, Y. Matsuno and M. Sato, "DS-Bench Toolset: Tools for Dependability Benchmarking with Simulation and Assurance", *42nd IEEE/IFIP International Conference on Dependable System and Networks (DSN 2012)*, 8 pages, June 2012.

[11] History of Scania, *http://www.Scania.com/Scania-group/history-of-Scania*

[12] I.Bate, T.Kelly, "Architecture Consideration in the Certification of Modular Systems. Reliability Engineering and System Safety," vol. 81, Issue 3, pp 303-324, 2003.

[13] I. Habli, R. Palin, R. Rivett and D. Ward, "ISO 26262 safety cases: compliance and assurance".

[14] I. Habli, T. Kelly, "A Safety Case Approach to Assuring Configurable Architectures of Safety-Critical Product Lines", *Proceedings of the International Symposium on Architecting Critical Systems (ISARCS), Prague, Czech Republic*, 2010.

[15] I. Habli, T. P. Kelly, "Process and Product Certification Arguments - Getting the Balance Right Workshop on Innovative Techniques for Certification of Embedded Systems", *Proceedings of 12th IEEE Real-Time and Embedded Technology and Applications Symposium, San Jose, California, United States*, 2006.

[16] IEC 61508:2010, "Functional safety of E/E programmable electronic safety-related systems."

[17] ISO26262, Functional safety for road vehicles, *International Standard*, November 2011.

[18] J. Johansson, N. Karlsson,"Gap analysis of Scania development of electric functionality and ISO 26262",*Malardalen University, School of Innovation, Design and Engineering, Master thesis*, 2011.

[19] Linkedin Group:,*http://www.linkedin.com/groups/ISO- 26262-Functional-Safety-2308567*

[20] M. Born, J. Favaro, and O. Kath, "Application of ISO DIS 26262 in practice", *Proceedings of the 1st Workshop on Critical Automotive applications: Robustness Safety*, 2010.

[21] M. Feather, L. Markosian, "Building a Safety Case for a Safety-Critical NASA Space Vehicle Software System", *Proceedings of the 4th IEEE International Conference on Space Mission Challenges for Information Technology*, 2011.

[22] N. Leveson, "White Paper on The Use of Safety Cases in Certification and Regulation", Journal of System Safety, updated May 6, 2012.

[23] P. Wallbeck, "Fuel Level Estimation for Heavy Vehicles Using a Kalman Filter", *Masters thesis performed at Division of Vehicular Systems, Department of Electrical Engineering, Linkoping University, 24 of November 2008.*

[24] R. Alexander, T. Kelly, Z. Kurd, J. McDermid, "Safety Cases for Advanced Control Software: Safety Case Patterns", Technical report, University of York, 2007.

[25] R. Dardar, B. Gallina, A. Johnsen, K. Lundqvist, M. Nyberg, "Industrial Experiences of Building a Safety Case in Compliance with ISO 26262", *Proceedings of the 2nd WoSoCER, joint event of the 23rd International Symposium on Software Reliability (ISSRE), Dallas (Texas), USA, 29 of November, 2012.*

[26] Safety Case definition from UK Defence Standard 00-56 Issue, http://www.dstan.mod.uk/standards/defstans/00/056/02000400.pdf.

[27] Scania Technical Product Data 1949329, Allocation Element Requirement AER Fuel Level Estimation: AE201.

[28] Scania Technical Product Data 1949330, "Allocation Element Requirement AER Low Fuel Level Warning: AE202".

[29] SESAMM model concept, *http://wiki.inline.Scania.com/wiki/SESAMM_model_concept.*

[30] S. Ali, M. Sulyman, "Applying Model Checking for Verifying Functional Requirements of a Scanias Vehicle Control System", *School of Innovation, Design and Engineering Malardalen University, Vasteras,Sweden, September 2012.*

[31] S. Amberkar, B. J. Czerny, J. G. DAmbrosio, J. D. Demerly and B. T. Murray, "A Comprehensive Hazard Analysis Technique for Safety-Critical Automotive Systems", *SAE 2001 World Congress, Detroit, Michigan, United States.*

[32] S. Friedenthal, A. Moore, R. Steiner, "A Practical Guide to SysML: Systems Modeling Language", *Morgan Kaufmann, 9780123852069, 2011.*

[33] T. P. Kelly, "Arguing Safety- A systematic Approach to Managing Safety Cases", *Department of Computer Science, University of York, UK. 1998.*

[34] T.Kelly, J. McDermid, "Safety Case Construction and Reuse Using Patterns", *Proceedings 16th International Conference on Computer Safety and Reliability, York, 1997.*

[35] T.Kelly, J. McDermid, "Safety case patterns-reusing successful arguments", *IEEE Colloquium on Understanding Patterns and Their Application to System Engineering, 1998.*

[36] T. Kelly, R. Weaver, "The Goal Structuring Notation A Safety Argument Notation", *Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases, 2004.*

[37] W.S. Greenwell, C.M. Holloway, J. C. Knight, J.J. Pease, "A Taxonomy of Fallacies in System Safety Arguments", *Proceedings of the 24th International System Safety Conference, 31 July - 4 August 2006, Albuquerque, New Mexico.*

[38] W.Chen, W. Li and H. Zhang,"Model-based Hazard Analysis Method on Automotive Programmable Electronic System", *3rd International Conference on Biomedical Engineering and Informatics (BMEI)*, 2010.

Appendices

A. Standard Deviation Analysis

This appendix shows the standard deviation for KF, exponential filter and the fuel level sensor for different driving scenarios. The following table and figure shows that KF has a lower standard deviation than the exponential filter except for for the fifth scenario because the latter had a stable initial state that was not affected by the changes in the sensor readings.

Table A.1: Standard deviation for both of the filters and the sensor in different driving scenarios [31]

	Driving scenario	**Length**	**STD KF**	**STD Exp. Filter**	**STD Sensor**
1	Driving mostly on highway.	2.8h	0.2713	0.4124	1.4515
2	Driving mostly on highway. (See Figure 3-12)	4.8h	0.3799	0.7295	1.4626
3	Driving on highway south of Södertälje, Sweden.	32min	0.1573	0.2446	1.8887
4	Driving with a small amount of fuel in the tank to see how the fuel movements affect the fuel level sensor. The vehicle is driven on a road known to contain certain road slopes and curves.	14.4min	0.1599	0.4327	2.8227
5	Similar scenario as nr 4.	8.7min	0.6991	0.5694	2.9378

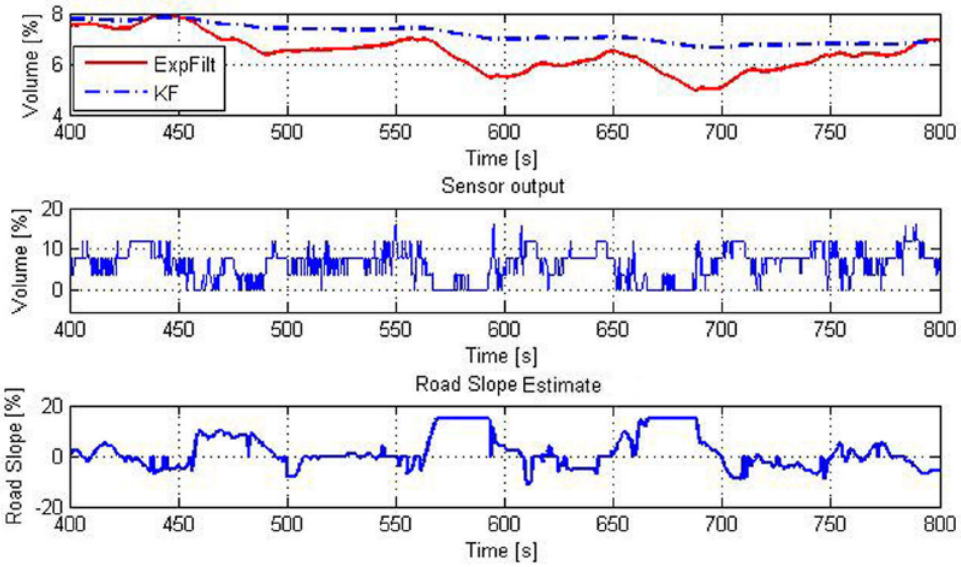


Figure A.1: Simulation of the standard deviation for both of the filters and the sensor [31]