Matthias Lamprecht, BSc.

Blockset for the Development of Safety Concepts in the Model-Based Development

Master Thesis

to achieve the university degree of

Diplom-Ingenieur

Master degree programme: Mechanical Engineering and Business Economics

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.-Ing. Dr.techn. Mario Hirz

Institute of Automotive Engineering

Dipl.-Ing. Dr.techn. Adam Schnellbach

AVL List GmbH

Graz, June 2018

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used.
The text document uploaded to TUGRAZonline is identical to the present master thesis.

_____	_____
Date	Signature

KURZFASSUNG

In der Fahrzeugindustrie ist man mit der Herausforderung konfrontiert, dass Fahrzeugentwicklungen umfangreicher und damit auch komplexer werden. Ein Ansatz um dieser Herausforderung in Entwicklungsprojekten erfolgreich zu begegnen, ist die modellbasierte Entwicklung, die in ihrem Einsatz allerdings noch problembehaftet ist. Ein Grund für den zunehmenden Entwicklungsaufwand sind die zahlreichen eingebetteten Systeme. Diese Systeme und das Auftreten von Gefährdungen im Zusammenhang mit diesen Systemen ist sicherheitsrelevant und daher ist deren Entwicklung in der ISO 26262 spezifiziert. Um Gefährdungen zu vermeiden werden Safety Goals bestimmt, die im Functional Safety Concept durch die Definition von Maßnahmen und Anforderungen weiter spezifiziert werden. Das Erstellen des Functional Safety Concepts folgt dabei ebenfalls dem Trend zu einem modellbasierten Ansatz, allerdings ist auch hier der Ansatz noch nicht ausgereift.

Die AVL ist ein Unternehmen, das im Bereich der Fahrzeugentwicklung tätig ist. Auch bei AVL wird zunehmend auf modellbasierte Entwicklungsansätze gesetzt. Die Maßnahmen und Anforderungen um Safety Goals zu erreichen sind aber zurzeit noch textbasiert und werden manuell erstellt.

Diese Masterarbeit trägt zur modellbasierten Entwicklung bei und soll den Grundstein legen um künftig das Erstellen von text-basierten Maßnahmen und Anforderungen zu vereinfachen. Dazu hat die AVL ein Functional Safety Concept eines Hybridfahrzeuges zur Verfügung gestellt. Im Functional Safety Concept wurde nach wiederkehrenden Mustern gesucht. Dabei wurden Muster im Aufbau und im Inhalt der Functional Safety Requirements entdeckt, deren Nachbildung auch mit Blöcken in einem Modell möglich ist. Es wurden daher sowohl in Matlab/Simulink als auch mit SysML in Integrity Modeler vordefinierte und wiederverwendbare Blöcke entwickelt und in einer Library gesammelt. Um zu zeigen, dass diese Blöcke auch zum Erstellen eines Modelles geeignet sind, wurden sämtliche Functional Safety Requirements eines Safety Goal in beiden Softwaretools erfolgreich modelliert.

Der entwickelte Ansatz liefert vielversprechende Ergebnisse um die Erstellung von textbasierten Functional Safety Requirements zu erleichtern. Die Anwendung des Ansatzes in Projekten erfordert jedoch noch weiterführende Entwicklungsarbeit.

ABSTRACT

The automotive industry faces the challenges of complex modern automotive development with model-based approaches, but struggles with their implementation. Some reasons for complexity are the numerous embedded systems of modern vehicles. The state of the art for the development of safety relevant automotive systems is addressed by the ISO 26262. The safety goals describe top level safety requirements and the functional safety concept breaks these requirements further down to functional safety requirements. The creation of functional safety concepts changes from manual to model-based approaches, but is still challenging.

AVL is a company in the automotive development sector. At AVL functional safety requirements are text-based and created mainly manually. To improve the development process of their functional safety concepts, the development is also pushed toward a model-based approach.

The contribution of this thesis lays the foundation for a simplified creation of text-based functional safety requirements by using a model-based approach. To this end, AVL provided the functional safety concept of an HEV, which was analyzed in this thesis for recurring patterns. The analysis revealed recurring patterns in the structure and content of the functional safety requirements. These patterns were identified to be reproducible by blocks in a model. The developed blocks were gathered in a library, which was developed in Matlab/Simulink and then transferred to SysML in PTC Integrity Modeler. The library contains all predefined and reusable blocks. As a proof of concept, the library was used successfully in both tools for the creation of functional safety requirements of a whole safety goal.

The library of blocks for the generation of text-based functional safety requirements from a model is a promising approach for the simplification of the functional safety concept development. To derive an executable model a more detailed elaboration is necessary.

TABLE OF CONTENT

1 INTRODUCTION

1.1 Motivation

Due to worldwide environmental problems, governments dictate increasingly strict emission standards. This led to the necessity of reduced fuel consumption and as a result, to the push in the development of new, or modified forms of propulsion systems. One of these developments is the hybrid powertrain, where, for example, a traditional internal combustion engine (ICE) is combined with a high voltage (HV) battery and an electric machine. [22]

To manage the functions of different vehicle components, electronic control units (ECUs) have been used since the 1970s. Such ECUs are embedded systems and control e.g. the power source, the brakes, the transmission and the infotainment of modern vehicles. [21], [22]

To fulfill these tasks, in modern premium cars roughly one Gigabyte of software code is implemented in approximately 100 ECUs [21]. This is four times more than a decade ago [1]. These embedded systems in the vehicle contain approximately 100 million lines of software code, which is 15 times more compared to the “Boing 787 Dreamliner” with about 6.5 million lines of code, and an indicator for the complexity of modern vehicles [29]. Moreover, 25 % of the costs of a vehicle are owed to these systems, and the complexity and portion of the costs is expected to grow even further [32]. These examples show the scope of work with which today’s engineers are confronted and the importance of overcoming this challenge.

One especially important task of the embedded systems is the management of the safety relevant systems of the vehicle. Due to the dependability of safety relevant systems from many specialized engineering fields, the automotive development approach of “separation of concern” is problematic, because cooperation between the engineering fields is necessary [21]. One promising approach to handle this challenge is model-based development [7], [21], [31].

Furthermore, safety-critical systems like electric and electronic systems in the automotive domain are regulated by the ISO 26262 standard. This standard adds additional complexity to the development of safety-critical systems. [17], [30]

The contribution of this thesis to automotive development includes the improvement of a part of the development of safety-critical systems by applying a model-based approach.

1.2 Modern Automotive Development

The automotive industry is currently facing challenges in several fields. The focus in this section is limited to new propulsion systems and the arising challenges due to the interplay of propulsion systems with the increasing number of embedded systems.

1.2.1 Model-Based Development

The highly competitive automotive market leads to the necessity for cost reduction and shorter development time. Furthermore, this is combined with the rising complexity of vehicles and their propulsion systems. The established original equipment manufacturers (OEMs) and their suppliers must adapt to be competitive on the market. Unlike experiment-based engineering, or the separation of concern, the development with a model-based approach allows the required adaptations and is therefore increasingly used in the automotive industry. [19], [31]

Model-based development helps to reduce the complexity of systems and allows interdisciplinary development [7]. A model is the abstraction of reality. It represents reality and contains the information and context information of it, but it covers only those parts of reality, which are estimated to be relevant in the context of the purpose [17]. Such models are the foundation of whole development processes. They support the determination of requirements, the design, the implementation and the test and verification process [31].

Hence, it is also applicable to the development of embedded systems.

Although it is a promising approach which should solve many of the problems of the modern automotive development processes, there are still issues which are not entirely solved. The cooperation of different engineering disciplines, the connection of different development tools and different abstraction levels are still not seamlessly connected and therefore cause problems in the development process. [21]

1.2.2 Hybrid Electric Vehicles

Hybrid electric vehicles (HEV) are vehicles which offer more than one energy storage system and more than one on-board energy converter system. Although other propulsion systems are possible, this section will only take the combination of ICEs and electrical motors in account. These HEVs can be classified in two ways. [19]

The first possibility for classification is by the capabilities of the electrical system:

- Micro-HEV
- Mild-HEV
- Full-HEV
- Plug-in HEV

[19]

The second possibility is the classification by the architecture of the drive train[1]:

- Series hybrid electric vehicle
- Parallel hybrid electric vehicle
- Series-parallel hybrid electric vehicle
- Complex hybrid electric vehicle

[19]

Further classifications are possible for the parallel hybrid, according to the position of the electrical machine along the drive shaft:

- Parallel-1 Hybrid Electric Vehicle (P1-HEV)
  (electrical machine between ICE and clutch)
- Parallel-2 Hybrid Electric Vehicle (P2-HEV)
  (electrical machine between clutch and transmission)
- Parallel-3 Hybrid Electric Vehicle (P3-HEV)
  (electrical machine between transmission and final drive)
- Parallel-4 Hybrid Electric Vehicle (P4-HEV)
  (electrical machine and ICE on independent drive axles)

[19]

These classifications show the diversity in hybrid systems. Nevertheless, all these powertrain systems share similar components and the necessity to be controlled.

On the following figure the HEV-powertrain architectures, according to the second classification possibility, are displayed schematically. All the shown powertrain architectures in this figure share similar components, but the arrangement of the different components differ significantly. All the architectures have in common that the energy storage system (fuel tank, battery) is the energy source for the on-board power converter (ICE, electric motor), which provides mechanical energy to the vehicle. Furthermore, the generator, the transmission and the power converter can be seen. Another

[1] The literature offers several possibilities for the classification in accordance to the architecture. This is one example.

interesting note is the connection of the different elements, which is either mechanical, hydraulic, or electrical.

An important annotation to the figure is the difference between the on-board power converter (e.g. electric motor), which changes energy from one physical form to another (e.g. chemical to mechanical energy) and the power converter, which is used to provide the suitable current and voltage for the electric motor [11].

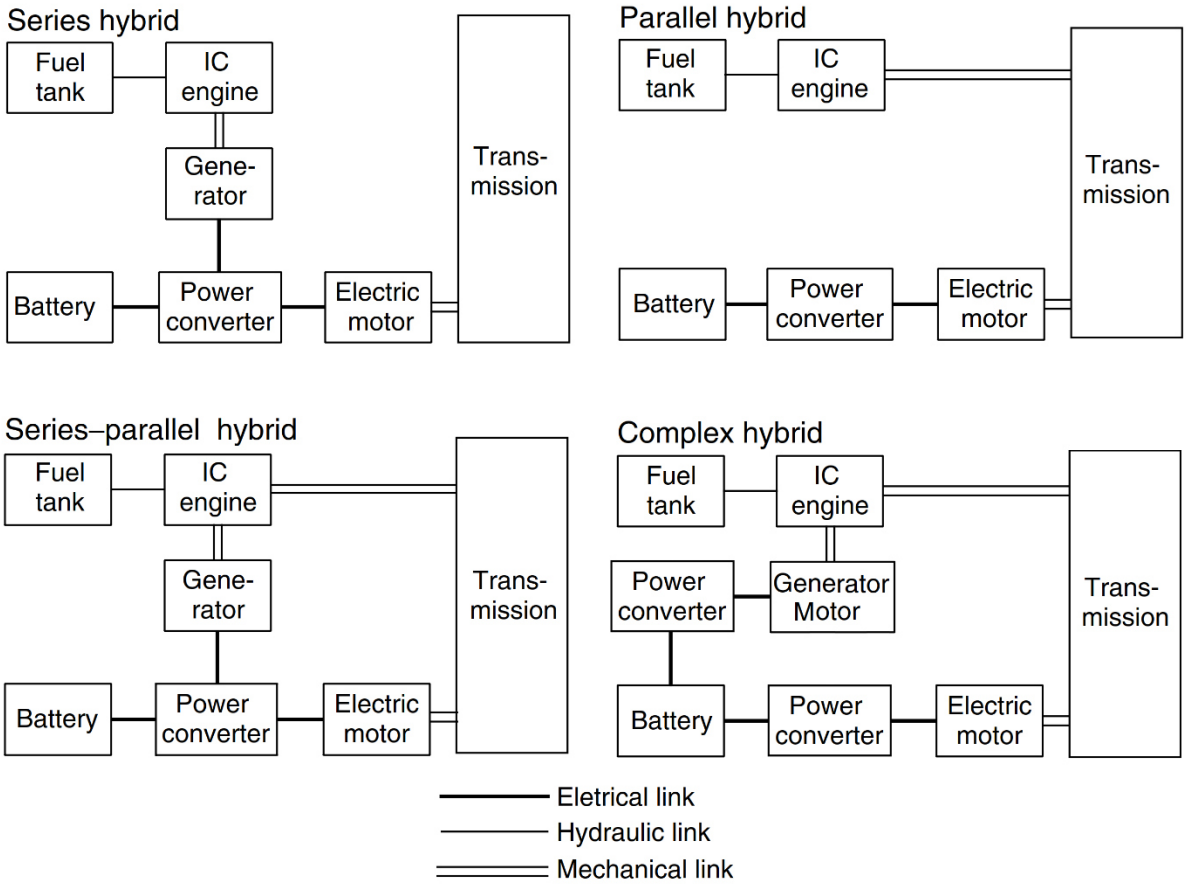

Figure 1.1: Exemplary classification of HEVs according to the drivetrain architecture, modified from [11]

1.2.3 Embedded Systems

The components of a powertrain and other components of a vehicle are controlled by embedded systems. The embedded systems include many control units, which are connected among themselves. The following schematic figure shows a simplified depiction of the components of a single-shaft torque combination of a powertrain. Since the electric machine is being implemented between transmission and clutch, the powertrain is classified as P2-HEV powertrain. Furthermore, the figure illustrates the control units, which handle the individual components of the powertrain, e.g. the Transmission Control Unit (TCU), the Engine Management System (EMS), the Clutch Control Unit (CCU), the Motor Control Unit (MCU) and the Battery Management System (BMS). Apart from electrical and mechanical

connections, which were illustrated in Figure 1.1, this figure also shows the electronic connections of the control units via a bus-system. The bus-system enables the communication between the different control units and the exchange of information with the Hybrid Control Unit (HCU). [22]

The HCU manages the HEVs control units. As an example, the driver demands a specific torque via the accelerator pedal position. The HCU receives this information and also considers other input information before commanding a specific torque request to the EMS and MCU. The EMS and MCU control the engine and electric machine with appropriate signals, respectively. [23]

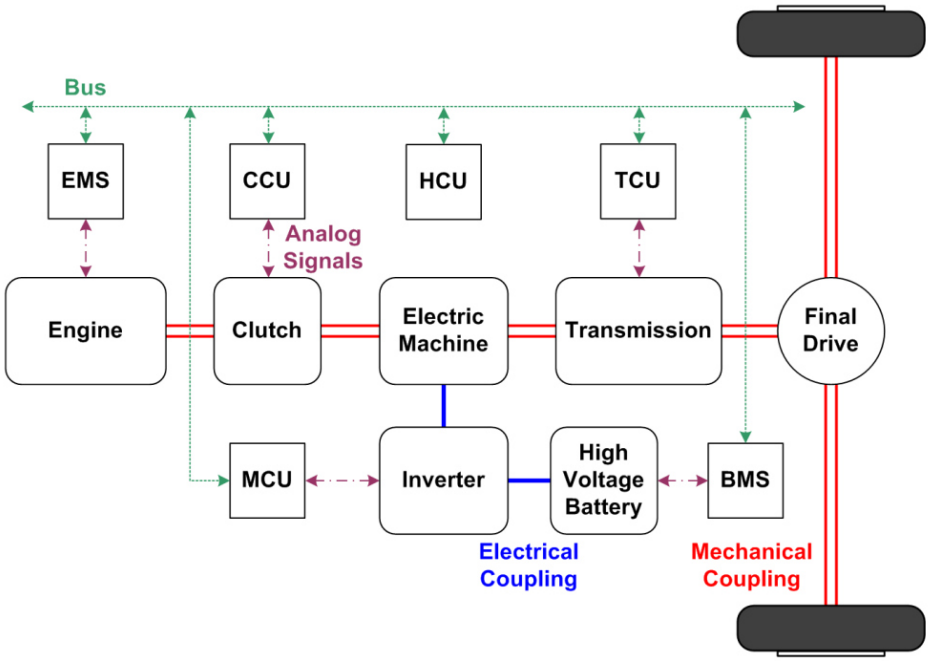

Figure 1.2: Components and controllers of a P2-HEV [22]

One problem of the embedded systems is the potential occurrence of functional failures, which could cause harm. For example, the high voltage battery can be overcharged and cause an explosion, which may injure humans. Another example is the unintended movement of the vehicle in front of a pedestrian crossing, due to unintended electric machine torque. Hence, the occurrence of such events must be prevented, making embedded systems safety-critical. [35]

1.2.4 Functional Safety

Because of the demand for electric, electronic and programmable electronic systems to perform safety-critical functions, the automotive industry released a standard to define the state of the art for the development of such systems in 2011. This standard is the ISO 26262 “Road vehicles – Functional safety”, an adaptation of the IEC 61508, and regulates electrical and/or electronic (E/E) systems for road vehicles weighing up to 3.5 tons and the automotive industry’s fulfillment of this standard. [22], [32]

Safety-critical systems, like the previous mentioned embedded systems, are in the scope of this standard. [22]

Functional safety means the development and application of measures to prevent threats and their causes, or reduce the risk of occurrence, below an acceptable limit. One section of the ISO 26262 defines the management of functional safety. The ISO 26262 specifies a safety life cycle to support and achieve functional safety. The safety life cycle includes the concept phase, the development phase as well as the production and operation phase of a product and defines methods, activities and necessary documentation, which must be established to fulfill the standard. [17], [26]

The ISO 26262 and the safety life cycle are only of secondary relevance for this thesis, but Figure 1.3 illustrates the rigorous effort necessary to fulfill this standard. The figure shows the complete life cycle and the different phases and tasks within the phases. Furthermore, the V-Model for the development phase and the V-Models for the hardware and software development process are hinted.

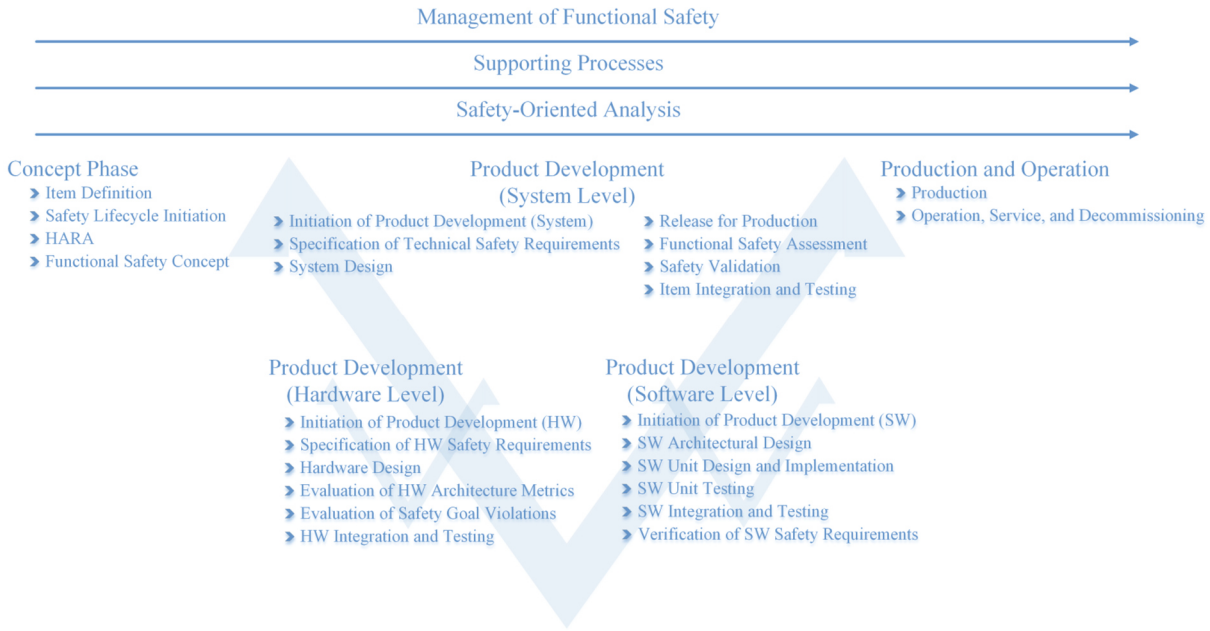

Figure 1.3: Product safety life cycle of E/E systems [32]

Due to the extensive scope of this standard including 687 requirements, 100 work products and 62 tables [20], the complexity and development effort increases [17], and the consistency and traceability of all development aspects is a major challenge [21].

This thesis aims to improve the development of the functional safety concept, which is a part of the concept phase. The other phases are not in the scope of the thesis.

1.2.5 Functional Safety Development in the Concept Phase

Following the ISO 26262 standard for vehicle development, a hazard analysis and risk assessment (HARA) is necessary [9], [27].

Hence, the first step in the concept phase is the gathering of information and the definition of functions and functional requirements of the planned vehicle. A vehicle function is for example the recuperative braking system. The according functional requirements are the circumstances for activation. [22]

Based on the gathered information and functions of the planned vehicle, possible malfunctions and the resulting hazardous events are determined. Under consideration of their criticality, each of the hazardous events is evaluated and assigned with an Automotive Safety Integrity Level (ASIL) from A to D, with ASIL A denoting the least stringent safety level. [9], [17], [27], [30]

If no safety relevance is given, the classification of the hazardous event is "QM" ("Quality management") [17].

The ASIL rating is influenced by the controllability, probability of exposure and severity of a hazardous event and a higher ASIL level requires more stringent methods of risk reduction. The hazardous events rated with an ASIL are assigned with a safety goal. [21], [22], [30], [35]

Each safety goal is assigned at least one safe state (SS) and maybe a degradation mode (DM), which should prevent the occurrence of the hazardous event. [22]

For example, a safety goal is the prevention of HV battery outgassing. In case the HV battery temperature reaches during charging a critical level were outgassing is possible, a SS is triggered, which interrupts the charging process.

The following table illustrates examples for hazardous events, their ASIL classification and the definition of corresponding safety goals of a two-motor hybrid powertrain.

Table 1.1: Example of a HARA from a two-motor hybrid powertrain, based on [9]

Hazardous Event	Safety goal	ASIL
Unintended acceleration	Unintended acceleration should be avoided	A
Unintended deceleration	Unintended deceleration should be avoided	B
Unintended vehicle start	Unintended vehicle start should be avoided	C
Unintended sideslip	Unintended vehicle sideslip should be avoided	B

The safety goals are the source for the definition of functional safety requirements of the embedded systems and controllers [5], [22]. Besides other information, the safety goals and requirements are gathered in the functional safety concept and allocated to elements. The functional safety requirements gathered in the functional safety concept are a part of the concept phase and hence, abstract, which means that they do not imply any technical specification [21]. Nevertheless, these requirements are the foundation for the elaboration of a technical safety concept, and the technical safety requirements [5], in the product development phase, which follows the concept phase [22].

The functional safety concept, as one of the necessary work products to meet the ISO 26262 standard, is the work product for the investigation of this thesis.

1.3 Objectives of the Thesis

This thesis shall lay a foundation in the field of functional safety to simplify and improve the development of functional safety concepts, using model-based development. On a long-term basis, the text-based elements of a functional safety concept shall be generated from a model. This task would exceed the scope of one thesis, hence, this thesis shall only lay a foundation.

The major task is the analysis of an existing text-based functional safety concept for recurring patterns and the development of a model-based equivalent of these patterns.

The potentially found patterns are the foundation for the development of a reusable model library of different blocks. The blocks shall be the representation of the text-based patterns of the functional safety concept in a model. The library shall be developed for the software tools PTC Integrity Modeler[2] (with OMG SysML[3]) and The MathWorks Matlab/Simulink[4]. Therefore, the functional safety concept of a HEV, which is still in the development stage, is provided by the AVL List GmbH[5] [6]. To get an out-of-the-box solution, no further inputs are provided by AVL.

To show the capabilities of the library blocks, they shall be used exemplarily on one safety goal. Moreover, the software tools shall be compared very briefly for their most conspicuous differences, from a user-friendliness perspective.

Not included in this thesis is the questioning of the exact origin of the content of the functional safety concept. The library shall not be ready to use in a project, or enable the development of an executable model with the possibility to simulate the modeled patterns, or produce any textual output information yet. Neither the whole functional safety concept shall be modeled, nor is the exact programming of the developed blocks, by filling them with code and connecting the blocks to a database in the field of view. Another boundary of the thesis is the modeling of CAN (Controller Area Network) communication, including the signals generation, processing, routing and time behavior, which shall only be modeled very rudimentarily. Furthermore, the software tool comparison shall not

[2] PTC Integrity Modeler will be abbreviated in future mentions with Integrity Modeler. A basic understanding of the software is assumed by the author.
[3] OMG SysML will be abbreviated with SysML. A basic understanding of the language semantics is assumed by the author.
[4] The MathWorks Matlab/Simulink will be abbreviated with Matlab or Simulink, respectively. A basic understanding of the software is assumed by the author.
[5] The AVL List GmbH (Anstalt für Verbrennungskraftmaschinen List) is a well-known company for developments in the automotive sector, especially for powertrain systems, test systems and instrumentation.
[6] In future mentions AVL List GmbH will be abbreviated with AVL.

include a complete overview of both tools and their capabilities, but give an insight into the differences for modeling with the tools in the everyday usage. Finally, a conclusion and outlook to future work shall be given.

1.4 Organization of the Thesis

This thesis is organized as follows: In Chapter 2, related work is discussed and in Chapter 3, the current situation and problems at AVL are examined. In the fourth chapter the method is described. In Chapter 5 the findings of the thesis are presented. An extensive example, which illustrates the findings, is presented in Chapter 6. Finally, Chapter 7 summarizes the thesis and provides an outlook to future work.

2 RELATED WORK

In this chapter publications are reviewed, which are related to the thesis' key topics "model-based development" and "functional safety development". The focus of this chapter lies on potential existing solutions and the described challenges within these two domains in order to work them into the thesis solution, rather than dealing with the specific published solutions, which are not applicable. For an outside the box view, publications from the AVL staff, even though from other departments, are excluded from this examination.

In Section 2.1 the approaches, which are made by the industry and academia, to meet the development challenges, especially of embedded systems, will be examined. How the implementation of safety is handled by the automotive industry is the matter of Section 2.2. Section 2.3 will draw relevant conclusions for the thesis.

2.1 Challenges of Modern Development Processes

In the year 1998, the work of [13] pointed out that the exchange of information with textual documents is problematic, because the communication of requirements to hardware and software developers can be ambiguous. The recommended solution for engineering domains is a model-based approach. This approach is considered to allow exact definitions of the architecture, inputs, outputs and behavior of the system. Other issues which can be solved with a model-based approach are the timeliness, consistency and traceability of system related information.

In [38] the model-based approach is applied to the development process of a continuously variable transmission (CVT). The issues of shorter time-to-market and functional diversity of the powertrain's controllers are highlighted as main reasons for a model-based approach. Similar to [13], the document-based development process is criticized, because OEM's textual dictation of specifications and their interpretation by the supplier can lead to problematic misunderstandings. Furthermore, the authors match the argumentation of Friedman in [15] that in document-based developments the detection of failures, which occur especially in early development phases, can only be identified during a later testing phase of the product, causing additional adaptation costs. [1], [14], [15] and [38] conclude that a model is favorable, because it supports the implementation of inputs and evaluation of outputs, without a mechanical or electronic test device and hence, a late failure detection can be avoided. The in [38] identified disadvantage of the model-based approach is the necessity for more personnel during the model creation, but this is balanced by the later effort reduction during testing.

The authors of [7] emphasize the importance of using models in early phases of the development process and criticize the work in isolated tools. It is pointed out that isolated tools hamper the reuse of models and consequently the full productivity potential of a model-based approach. In their survey the authors detect the common use of textual descriptions and the problem of model consistency and a missing semantic foundation. Furthermore, their survey reveals that communication between engineers, also from other departments, is improved with models and that OEMs are forcing suppliers to work in a model-based way.

Giese [16] also points out the importance of early model implementation in the automotive development process and describes how models are tailored to specific development purposes. Common modeling tools in the automotive domain are SysML, for system design with blocks and AUTOSAR, for software development. The authors emphasize the importance of consistency, when information or requirements, respectively, are exchanged between different development domains, models or tools. The exchange must often be ensured manually and is considered as a problematic process.

In the publication [10] of D'Ambrosio and Soremekun the problems of consistency are pointed out. Many documents are stored on different engineer's computers without access for other engineers. As a result, the information source is unclear and important information can be lost. The automotive industry faces this challenge by the implementation of SysML, which due to the necessary communication, also improves the cross-domain development. Other mentioned advantages of models compared to documents are the potential reuse of patterns and the connection of the model to databases, which allows automatic updates of requirements. The potential of reusability is especially stressed for safety related work, because it supports faster development processes and allows focus on the most critical safety issues.

The authors of [12] and [34] conclude that tools like Matlab and Simulink support the development process in a specific development phase, but lack in architectural design. Other development tools are necessary to gain all aspects of the design process of embedded systems. In both publications modeling languages like UML, SysML or EAST-ADL are mentioned to be favorable for architectural design, because these languages support different diagram types for different purposes.

The publication of Fotso and Rettberg, [14], recommends (similar to the publications of [38] and [1]) the use of a model-based approach to identify issues early in the system development, which allows

cost reductions and accelerated development. For high abstraction levels, SysML is preferred over UML because of its improved support for different engineering disciplines. Furthermore, exact notations are considered essential for larger projects. For simulation tasks, Matlab and Simulink are proven tools to reduce the development duration.

In [37] the authors conclude that UML as a modeling language is widely spread in the modeling domain because of its extensive support of the development process of hardware and software. Being an official modeling language is not the case for Simulink, but due to the tools widely spreading [6], the authors of [37] conclude that the standard blocks and functions of Simulink are also a de facto language standard. Furthermore, the author emphasizes the importance of libraries for the modeling process, because many developments of earlier projects can be adapted or reused for new projects.

2.2 Safety Development

The publication [17] of Hillenbrand describes the ISO 26262 and the challenges of meeting the standard. In addition to the necessary documentation and the cross-domain challenges, the standard is very influential in the concept phase of the system development, but stays very abstract and only gives minor guidelines for the realization. An attempt for safety-related development is the model-based approach. In the publication UML and the related languages SysML and EAST-ADL are described as potential modeling languages to support the model-based approach.

The authors of [8] propose the usage of models for safety-related developments, because document-based approaches are cumbersome in these multi-domain developments and lead to misunderstandings. It is argued that a model-based approach solves this issue, because it allows system specifications from different viewpoints and linkages on different abstraction levels. Furthermore, the necessary effort for functional safety development is often underestimated, which leads to project delays and additional costs. The additional advantage of models is considered in the improved overall view of the project's progress and the easier proof of the achieved safety. Moreover, with a model-based approach the adaptation and reuse of previous safety developments is possible. Nevertheless, are for a model-based approach the maintenance of dependencies of the different model layers and domain specific attributes considered as crucial and challenging.

For the reuse of previous embedded system development projects in the safety domain, the authors of [25] emphasize the necessity of providing evidence of the validity of these developments for the new system. A survey in the automotive, heavy machinery and avionics industry revealed that industry

is pushing ahead with reuse approaches. Nevertheless, the industry is considered to be lacking in the use of model-based development, systematic approaches to implement safety development processes in the development culture and the analysis of reuse impacts. It is concluded that a more systematic approach would support reuse strategies and hence, allow the industry to reduce costs and the time to market.

In [28] the authors consider modeling tools like Matlab and Simulink for simulation and SysML for the modeling of system architectures to be widely implemented in the industry. The exchange between different development domains is still a crucial task. Especially the domain of safety development, expected to follow the trend of model-based development, is facing several challenges such as lack in productivity, increasing complexity and cross-domain communication. Therefore, the authors see potential productivity and cost improvements in the safety domain by the reuse of previous developments and an improved variant management.

In [18] the authors emphasize the importance for companies fulfilling standards like the ISO 26262 and work with mature models as a market advantage for suppliers. Due to a various number of development domains with different abstraction levels, different tools and models, the connection and exchange of these domains is argued to be challenging. The transfer of natural language, for requirement descriptions, to models, is often unclear and their traceability between different models is an issue. The authors propose the usage of controlled natural language for the functional safety requirements domain. This approach restricts the natural language to specific patterns, which combine two advantages. The requirements are understandable for layperson and due to its structure, automatic processing in different models is possible.
This publication addresses the goal of the thesis, but the process is reversed. A model is created from text. Hence, the desired goal of simplification is not achieved.

The authors of [2] see a lack of traceability and consistency between the system architecture and the development of safety concepts. The safety concepts are often natural text, spreadsheets or requirement databases, which is considered problematic because updates of these safety concepts are time-consuming. The authors developed approach is similar to [18]. Based on model-based formalization, textual template artifacts are provided to the developer for the description of the safety concept in a more formal way. The derived requirements are allocated to the architectural elements and improve traceability.

Superficially, this approach is similar to the desired solution of the thesis, but the authors approach lacks in the creation of a simple model as foundation.

In [26] the authors criticize document-centric development in the field of functional safety development and the lack of guidelines for the fulfillment of the ISO 26262 standard. The authors state that minor changes of the system in a document-centric functional safety development require the update of many documents. Their proposal to improve the development process is the use of a model-based approach. The authors state that a model allows the creation of a consistent single information source for all system specifications. Moreover, analysis from different viewpoints and the creation of documents from the model are possible. To put this to practice, the usage of SysML is proposed. Furthermore, for the exemplary development process of an HEV, the authors introduce “integration levels”, which should clear up the functional interaction of different elements on various abstraction levels. This approach corresponds with the suggestions of [8].
The corresponding figure is 2.1, where one can see three levels. Level 0 is representing the whole vehicle, Level 1 represents components of this vehicle, like the hybrid powertrain and the chassis. In Level 2 the elements of the Level 1 components, like the HCU or the battery, are represented.

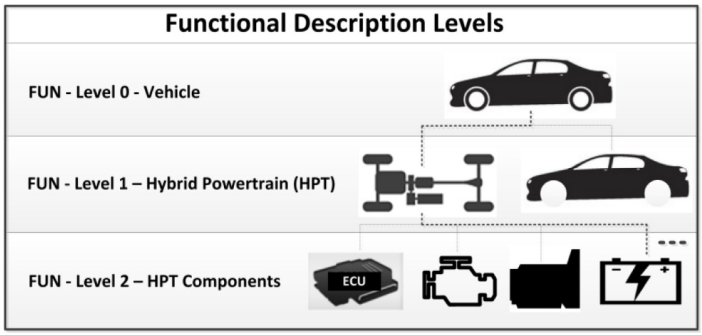

Figure 2.1: Functional levels of an HEV [26]

The thesis’ relevant statements of Sari and Reuss [29], are an increase of complexity in the development of embedded systems, due to the influence of multiple engineering disciplines and the proposal of model-based approaches to cope with this challenge. The creation of different abstraction levels, similar to [26], is considered to be especially helpful, because it allows early and level-specific analysis of the system. For this purpose, UML, SysML and EAST-ADL are applicable.
The following figure shows the authors’ multi-level approach with EAST-ADL regarding safety development activities. Every level of the model-approach supports different development tasks. At vehicle level, HARA and safety goals shall determine the necessary development tasks. The analysis level specifies the necessary development tasks to a concrete system architecture of functions and elements. The design level specifies these tasks even further to tangible hardware and software

functions and their connection. Finally, the lowest level defines the concrete software architecture, which is realized in AUTOSAR.

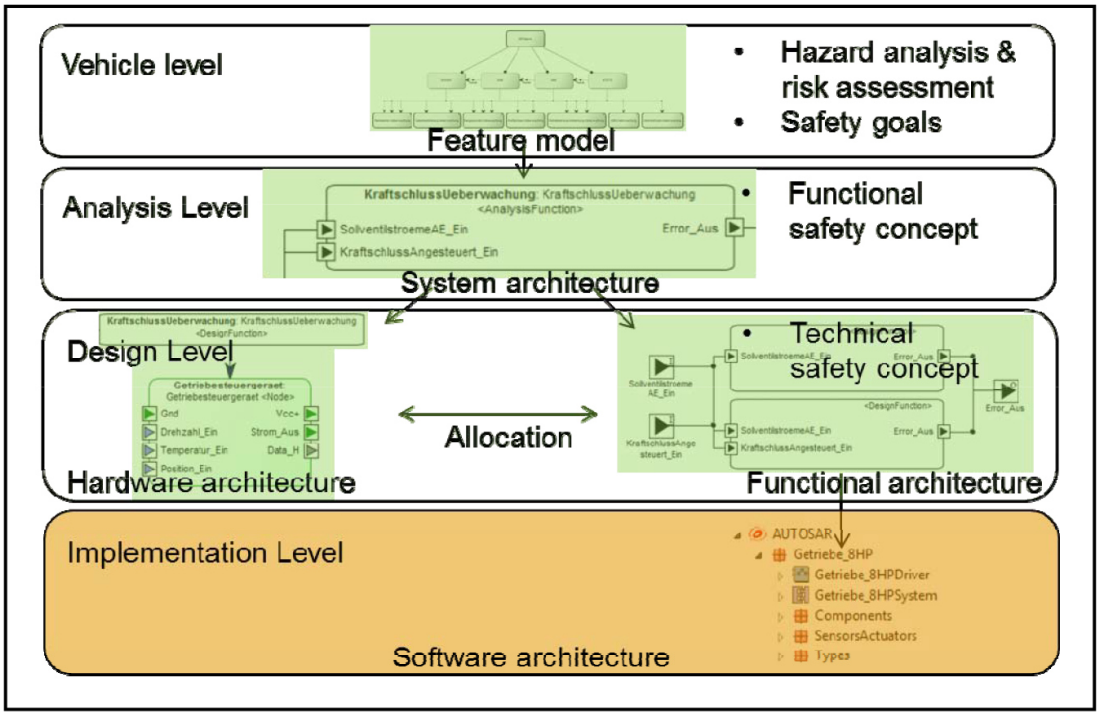

Figure 2.2: The levels of the EAST-ADL model and related safety activities [29]

The authors conclude that traceability, consistence and automatic document generation are the major advantages of this development approach. The approach also promises the automatic creation of a functional safety concept. This would be relevant for this thesis, but the authors give no comprehensible information about the specific creation of requirements.

2.3 Potentials for Improvement

Based on the reviewed literature of Section 2.1 and 2.2 a summary can be given and a conclusion can be drawn.

The literature research only revealed a single comparable solution related to the thesis goals. Furthermore, many challenges are identified in the field of automotive development, which should be taken into consideration.

Sari and Reuss [29] describe an approach with EAST-ADL as modeling language, to support a wide range of safety-relevant development aspects. One of the described functionalities of their approach is the automatic creation of a functional safety concept from a model. The publication does not reveal any details. Therefore, for this thesis decent conclusions regarding a potential adaptation in SysML and Simulink cannot be drawn.

The rest of the publication's tenor recommends the change from document-based to model-based approaches. This allows improved cross-domain cooperation on different abstraction levels, with consistent information from one single information source, less misunderstanding and more traceability. Furthermore, the connection of many different tools and the reuse and adaptation of earlier developments is possible. As a result, an overall improved development process is achieved.
For the model creation, languages like UML, SysML and EAST-ADL are widely spread and Matlab and Simulink also seem to be well accepted.
The ISO 26262 causes new challenges in the development process, which should also be manageable with model-based approaches.
Although many challenges are identified and seem to be well known, many papers reveal that textual descriptions and document-based development are still widely spread. A gap of information exchange and communication still exists in the cooperation between different abstraction levels and domains, and the reuse of former developments and automatic creation of output are still crucial. Hence, there is still potential for improvement in the development process of functional safety. The goal of the thesis is the contribution of such an improvement.

3 CURRENT SITUATION AT AVL

One of the business areas of AVL is the powertrain development. Within this domain, the functional safety development in accordance to the ISO 26262 standard is one sub-domain. The relevant parts of AVL's existing safety development approach are presented in this chapter.

In the first section the vehicle architecture is described with respect to safety developments. In Section 3.2 the safety development process at AVL is explained. The functional safety concept of a P2-HEV, on which this thesis is based, is investigated in Section 3.3. Finally, in Section 3.4 the problems AVL is confronted with are discussed and in Section 3.5 demands on the solution are defined.

3.1 The Vehicle Architecture at AVL

For the vehicle development process AVL uses a model-based approach and structures the model in several architectural levels. Each of these levels represents a different level of vehicle abstraction. The structure is supposed to support the reduction of complexity of the vehicle and back the development process by increasing the common understanding between different development domains and teams. It shall help to define the development boundaries for teams and the necessary interfaces to connect different domains in the horizontal and vertical model architecture. Moreover, it shall allow traceability along the whole development process.

In Figure 3.1 these architecture levels are shown. The levels 0 to 2 are mainly based on demands and requirements of the customer, whereas level 3 to 5 are derived from the development process within the AVL.

On the highest abstraction level, the whole vehicle is shown. On this level requirements like the interaction between vehicle and driver or functionalities and capabilities of the vehicle are defined. For example, a function would be the drivers' acceleration pedal push and the following acceleration of the car. An example for a requirement would be that the vehicle reaches a maximum speed of 130 km/h with a payload of 200 kg.

On level 1 the requirements and functionality of the vehicles subsystems are defined. An example would be the definition of attributes and functionalities of the powertrain. A functionality of the powertrain is for example the provision of torque to the system when it is demanded.

The specific components and their requirements are defined on level 2. In case of an HEV powertrain these are for example ICE, HV battery, transmission and electric motor. A requirement would be that the electric motor must provide at least 75 Nm torque during specific conditions.

Level 3 defines the specifications and requirements of mechanical and hydraulic elements and control systems. For example, the HCU as central control unit meets the driver's wish and allocates control-signals to other control units, which results in the vehicle's acceleration.

Following the controllers and putting mechanical and hydraulic elements aside, specifies level 4 the functionality of the control units. In case of the HCU is this for example the determination of the required torque.

In the safety development domain contains this level the functional safety requirements which are necessary to fulfill the safety goals.

Finally, Level 5 contains the code which enables the control units to fulfill their tasks.

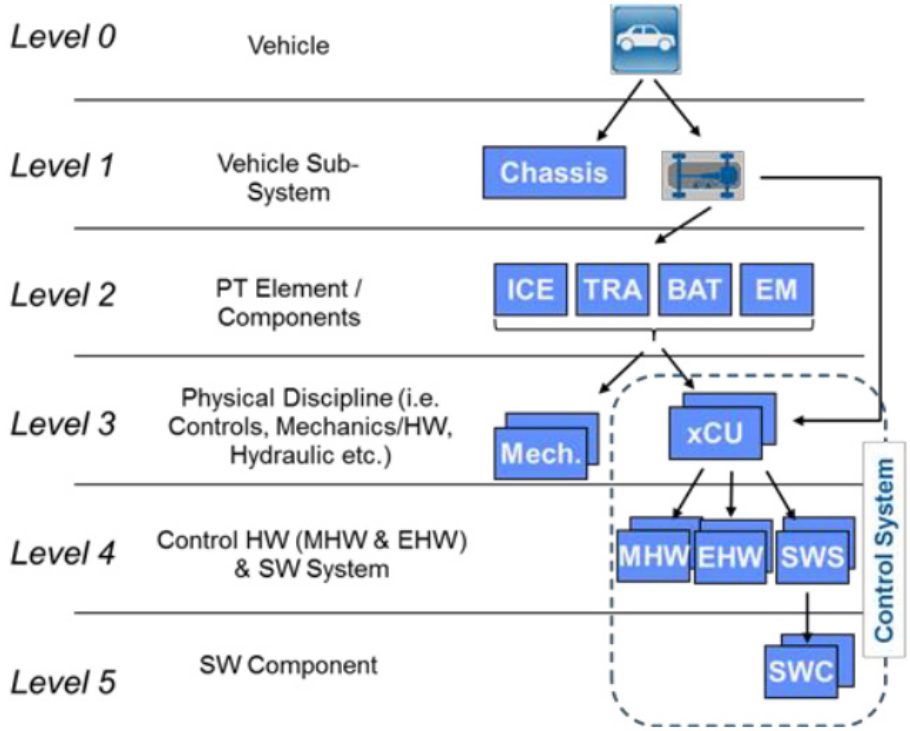

Figure 3.1: The architecture levels of vehicles at AVL [3]

3.2 The Safety Development Process at AVL

The safety development process at AVL follows the recommendations of the ISO 26262 standard. The development process is mainly model-based and SysML in Integrity Modeler is the software foundation for the architecture and behavior modeling. In this section the development process is outlined briefly for an HEV, without going into the details of any additionally used tools by AVL.

The first step in the development process is the item definition phase. Therefore, the HEV powertrain structure and behavior is modeled. This starts with the definition of system boundaries, like the connection to the rest of the vehicle via e.g. the communication interface or mechanical elements.

Afterwards the structure of E/E components of the powertrain are defined by AVL, including elements like the HV battery and the control units. [24]

The behavioral modeling defines functions of the HEV and the contribution of the E/E systems to fulfill these functions. An example would be the recuperation of energy and the involved components. [24] From the vehicle architecture perspective of Section 3.1, this compromises most parts of the levels 0 to 3.

The HARA follows the item definition phase. The goal of the HARA is the determination of safety goals. Therefore, the previously defined E/E system, its functions, requirements and the system boundaries are examined. The analyzation's output are potential malfunctions and hazards, like the outgassing of the HV battery caused by overcharging. The hazards are analyzed under the consideration of common vehicle operation situations and rated with an ASIL. For hazardous events with an ASIL rating higher than "QM", like the outgassing of the battery, safety goals are defined. The safety goals are top-level safety requirements from which further safety requirements are derived. The violation of safety goals may result in hazardous events and shall therefore be avoided. [22], [24]

Once the safety goals are determined the functional safety concept and its functional safety requirements are developed. Therefore, the energy and information flow in the system is examined for potential failures, faults and following failures. The results of this analysis are further investigated with FMEAs (Failure Mode and Effects Analysis) and FTAs (Fault Tree Analysis). From the results of these investigations, the element safety mechanisms (ESM) of the elements (e.g. HCU, OBC etc.) are derived and expressed with functional safety requirements[7]. The aim of the ESMs is the avoidance of safety goal violations by monitoring the system and set measures when necessary. An example would be the monitoring of the HV battery temperature. In case of crossing a defined temperature limit during HV battery charging, the appropriate SS is triggered and the charging is interrupted. [24]

From the functional safety concept are the technical safety concept and further specifications derived which are not included in this thesis.

3.3 The Functional Safety Concept at AVL

The functional safety concept of an HEV[8] which is developed by AVL is the foundation of this thesis. At AVL the functional safety concept is a document which is a gathering of the relevant measurements to fulfill the requirements of the ISO 26262 standard.

[7] To improve the readability, the term "requirement" will be used as a synonym for "functional safety requirement". If necessary for clarity, the term "functional safety requirement" will be used, instead of "requirement".

[8] The functional safety concept contains outdated diagrams of an earlier development stage, which only partly match the textual information of the document.

The whole functional safety concept of the HEV is organized in several sections of the document. An overview will be given in this paragraph and parts which are relevant for the thesis will be described in detail in the following sections.

The “Introduction” and the “Safety Goals” sections of the functional safety concept provide general information about the stage of development, abbreviations, general definitions for functional safety requirements, a system description and an overview of all safety goals.

The general definitions for functional safety requirements are described in Section 3.3.1 and the system description is shown in Section 3.3.2 of this thesis. In Section 3.3.3 three safety goals which were used in this thesis are presented.

The “Functional Safety Requirements” section of the functional safety concept contains information about mechanisms to prevent safety goal violations and will be described in Section 3.3.4.

Another relevant section (“Communication Matrix”) defines the specifications for the CAN communication between different elements. The specifications determine that the communication must be e.g. end to end (E2E) protected and shall include functions such as a cyclic redundancy check (CRC). The specifics of the CAN communication are not further discussed, because they are not in the focus of this thesis.

Further sections of the functional safety concept contain diagrams with Goal Structure Notation (GSN) as well as FTAs; these items are however of secondary relevance for this thesis and will not be further discussed.

3.3.1 Definitions for functional safety requirements

The “Introduction” section of the functional safety concept defines amongst other things that functional safety requirements are created in natural language with standardized “keywords” and so called “requirement sentence templates” with defined wording semantics. To prevent misunderstandings, the functional safety concept contains in the introduction a definition and interpretation.

Two good examples for the keywords and their description are shown in the following table.

Table 3.1: Examples of keywords used by AVL [4]

Term	Description
SHALL:	This word, or the terms "REQUIRED", means that the text describes an absolute requirement (shall be used for Health and Safety critical items).
SHOULD:	This word or the adjective "RECOMMENDED" in the text means that the requirement described has lower priority than the requirement described with "SHALL”.

The requirement pattern templates are used as a foundation for the standardized creation of functional safety requirements, and implement the mentioned keywords and wording semantics in their structure. Eleven templates are used for the description of these functional safety requirements. The templates 02 and 03 are shown as examples in Table 3.2. One can see the description of their purpose, structure and interpretation. Moreover, it shows typical keywords and semantics like “if”, “of”, “then”, “shall” and “within”.

In Section 3.3.4 are two functional safety requirements presented, where these templates are used.

Table 3.2: Examples of requirement pattern templates used by AVL, based on [4]

Template No.	Purpose	Structure	Interpretation
2	Fault detection and reaction without degradation	[02] IF §1 condition 1§ THEN the §2 element 2§ shall trigger §3 safe state(s) and/or degradation mode(s) 3§ within §4 time 4§	If the condition defined in text field #1 is fulfilled, the element in text field #2 shall trigger the safe state(s) and/or the degradation mode(s) in text field #3. The time between the condition being fulfilled for the first time and the fault being set shall not exceed the time defined in text field #4.
3	Degradation modes and safe states	[03] IF §1 degradation mode or safe state name 1§ is triggered THEN §2 element 2§ shall §3 perform action 3§ within §4 time 4§ {until restart; as long as the triggering fault is present}.	The element in text field #2 shall perform the action defined in text field #3 if the degradation mode or safe state, defined in text field #1, is triggered. The action defined in text field #3 shall be performed: - Either until restart, i.e. until the next ignition cycle; OR - as long as the triggering fault is present. The time between the triggering and the action (defined in text field #3) being finished shall not exceed the time defined in text field #4.

3.3.2 The Powertrain System Architecture

An important element of AVL’s functional safety concept is the description of the powertrain system architecture. It gives an overview of the powertrain and related elements. At AVL this is modeled in a SysML diagram which represents the structure and connection of all elements. This includes ICE, transmission, HV battery etc. and the necessary control units[9]. Furthermore, the connections for mechanical, electrical, material and information exchange are shown in this model.

[9] An extensive list of all abbreviations can be found at the end of the thesis.

The following figure shows the corresponding model of a P2-HEV. For example, in this figure one can see the powertrain and its elements within a dashed rectangular boundary and the HV-Plug outside the rectangle. The HV-Plug is connected to the On-Board Charger (OBC) via an information and an HV connection. The arrows of these connections indicate the possible flow directions. One can also see the CAN-Bus system and the central role of the HCU.

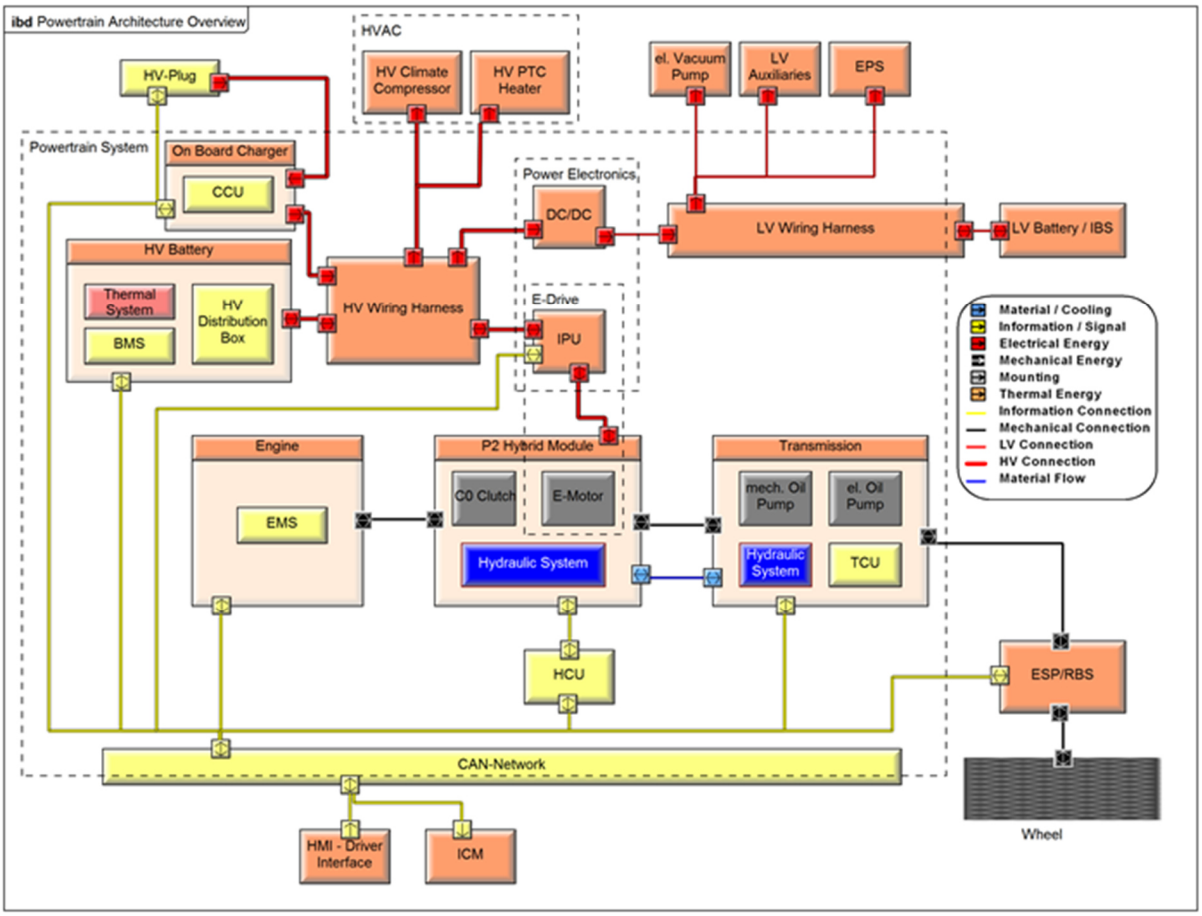

Figure 3.2: Powertrain system architecture model [4]

3.3.3 Safety Goals

Safety goals are one of the central content of the functional safety concept. In case of the examined P2-HEV the functional safety concept defines eleven safety goals. Three exemplary safety goals are listed in Table 3.3.

The motivation for safety goal 07 (SG07) is to ensure that the HV voltage level of all involved elements is reduced to an uncritical level in order to prevent the human life from high-voltage incidents.

The measures of SG08 shall ensure that the HV battery conditions remain within their allowed limit such that no outgassing hazards occur.

The protection of units supplied by the LV Bus from damage caused by overvoltage and mitigate the risks in case of undervoltage, is the target of SG09.

Table 3.3: Exemplary safety goals of the P2-HEV, based on [4]

No.	Description
07	HV_SG_07: Avoid unintended HV
08	HV_SG_08 Avoid HV battery outgassing
09	LV_SG_09: Avoid sudden loss of LV system supply

3.3.4 Safety Mechanisms and Functional Safety Requirements

The safety mechanisms and their specification with functional safety requirements are an essential part of the functional safety concept and represent the mechanisms to prevent safety goal violations. The safety mechanisms of AVL's functional safety concept are broken down into powertrain safety mechanisms (PTSM) and element safety mechanisms (ESM).

On powertrain level and element level the description of functional safety requirements is based on requirement pattern templates.

In addition to the PTSM, SysML activity diagrams, like the one shown in Figure 3.3, give information about the involved elements, ESMs and safety relevant functions (SRF) and their connection on powertrain level to prevent the specific safety goal violation. SRFs are closely related to ESMs and also described with functional safety requirements. Figure 3.3 shows a SysML activity diagram with its Electric Drive (EDRV, E-Drive) and HV battery (HVBATT), their connection and inputs, which are processed in ESMs and SRFs, respectively.

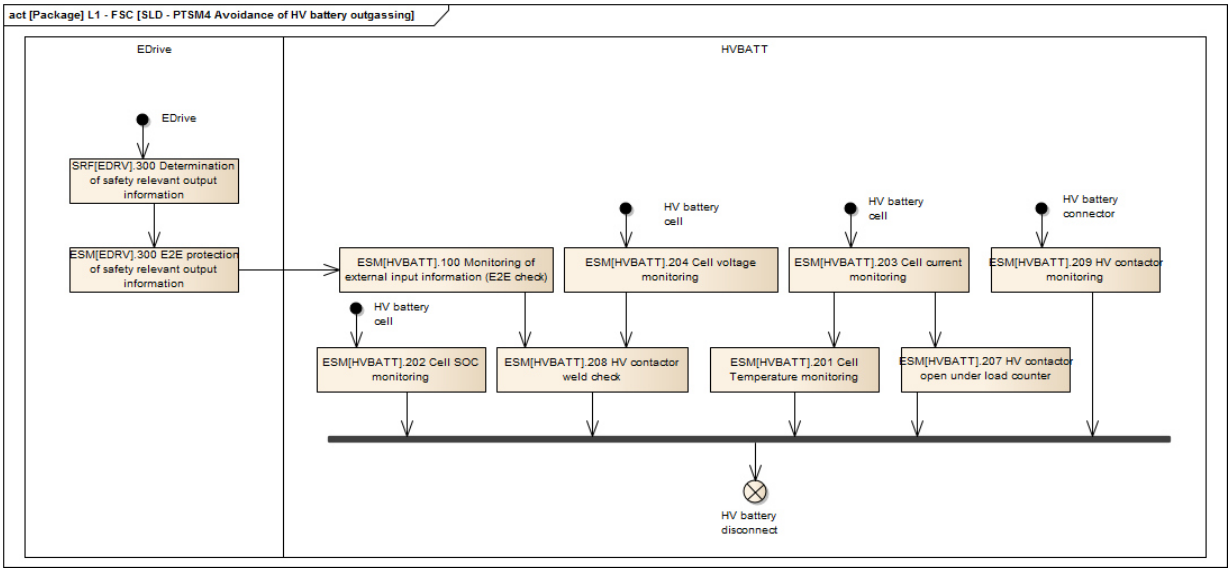

Figure 3.3: Activity diagram of PTSM4 – Avoidance of HV battery outgassing [4]

On element level of the functional safety concept, Internal Block Diagrams provide information about the interaction of the elements and the connections between the ESMs, SRFs, SSs and DMs of the element. Figure 3.4 shows the Internal Block Diagram of the PEPS. One can see SRFs, ESMs, their connections and the flow directions (arrows). [10]

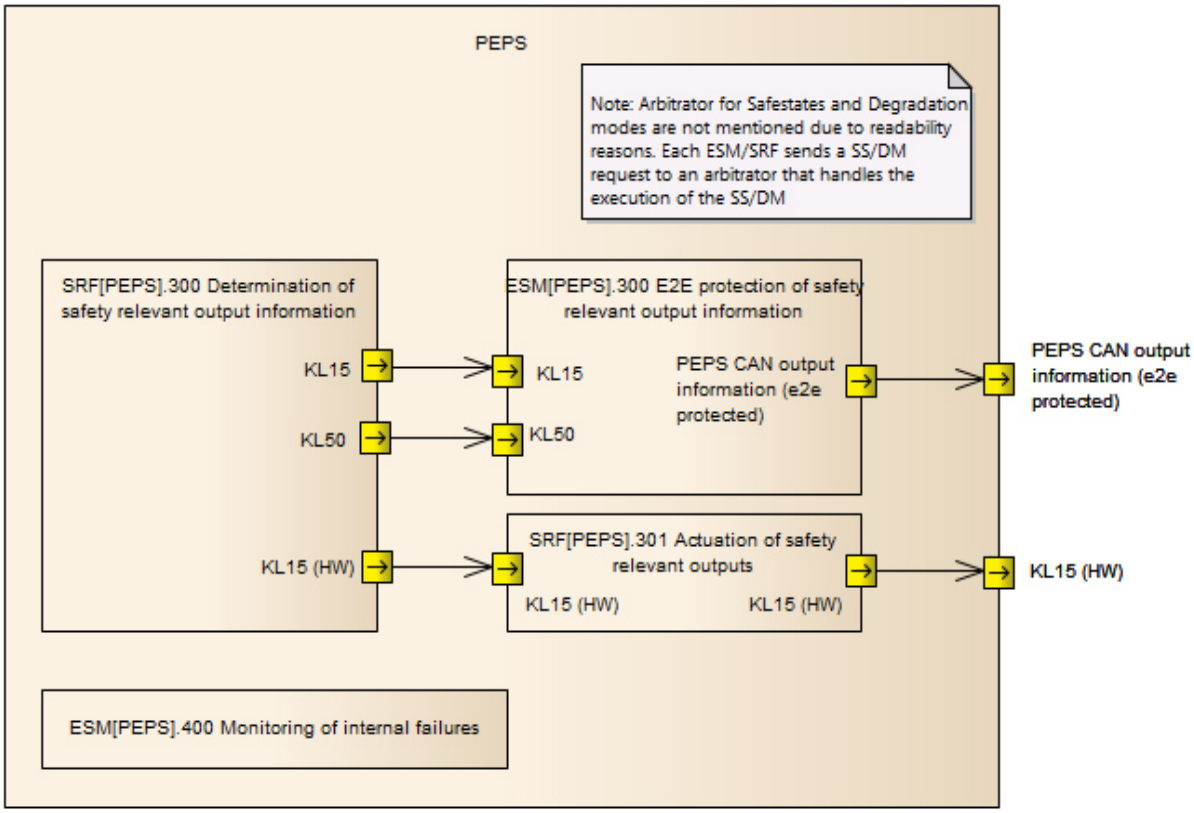

Figure 3.4: Example of the Internal Block Diagram of the PEPS [4]

The prevention of safety goal violations on element level are described in AVL's functional safety concept by ESMs, SRFs, DMs and SSs and contain one or several functional safety requirements. Unlike the functional safety requirements of ESMs, which monitor the system, DMs and SSs set active measures as reaction to failures detected by ESMs. These measures are usually initiated by signals from ESMs.

An example of such a requirement, based on a type two requirement sentence template, is shown in Figure 3.5. The functional safety requirement monitors if a crash occurs and triggers two SSs of

[10] During the work on this thesis, AVL replaced for functional safety developments the SysML modeling software and the Internal Block Diagrams. The influence on the thesis is negligible, but it can cause contradiction. The provided functional safety concept contained outdated figures, which are presented in the thesis, because they were the information basis. Figure 3.3, 3.4, 4.1 and 4.2 are part of the old developments. The findings of this thesis are created in the new software and with Activity Diagrams.

the EDRV when necessary. Moreover, the functional safety requirement defines in curly brackets the allowed duration for the fulfillment of the requirement.

The SS[EDRV].02 is shown in Figure 3.6. According to the SS shall the current not being fed back to the HV system.

Apart from textual description one could see the “REQ-ID”, which assigns every requirement with a specific identification index. Moreover, the status of the requirement, the ASIL rating etc. are information, which is visible in every functional safety requirement.

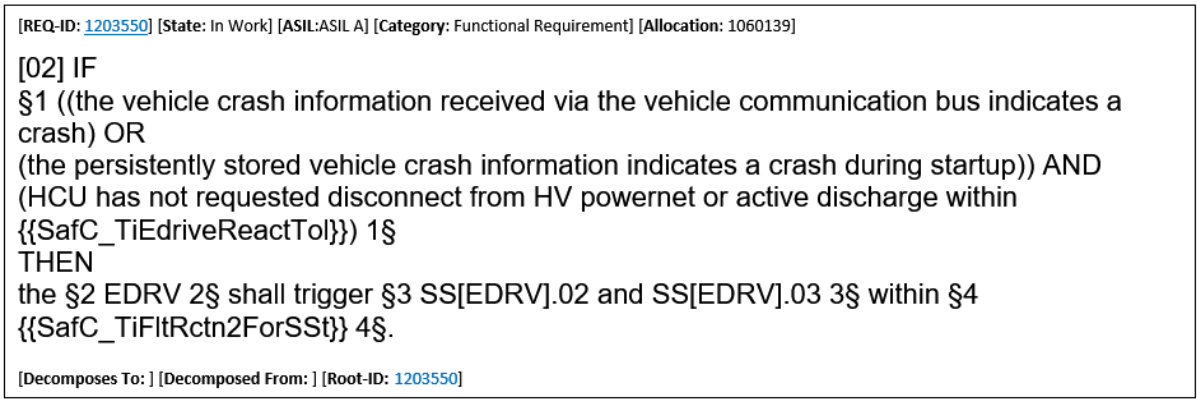

Figure 3.5: Example of a type two functional safety requirement [4]

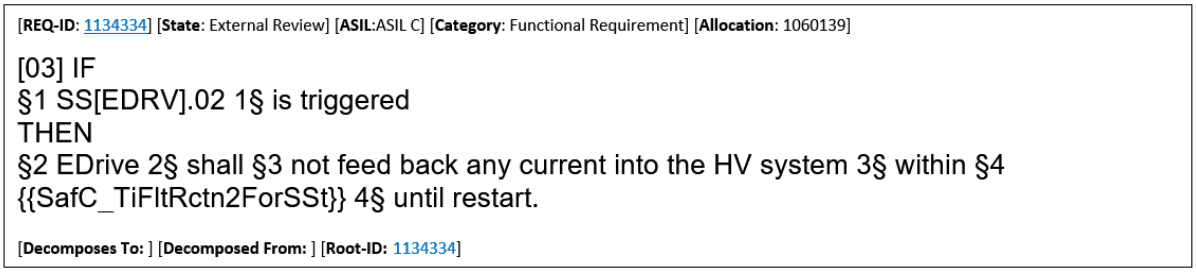

Figure 3.6: Example of requirement for a SS [4]

3.4 Problems

In the reviewed literature of Chapter 2, some publications recommend the use of natural language with predefined patterns and a model-based approach to support the development process. As the functional safety concept of AVL reveals, both recommendations are partly implemented in AVL’s development process. Nevertheless, the development process of meeting the ISO 26262 standard is still problematic. The following sections examine the most sustained problems, which occur in the functional safety development process at AVL and illustrates these problems with examples from the examined functional safety concept.

3.4.1 Consistency of Information

Although the development process at AVL utilizes a model-based approach, the functional safety requirements are derived manually in documents. The missing link between models and documents imply risks that important information, like changes in the model, are not implemented in the documents or differ between each document and/or model.

The worst-case scenario of faulty consistency is the creation and implementation of defective software, which causes harm to the end consumer. Barely less problematic is the identification of inconsistencies during the testing and implementation phase, because this causes adjustments, delays and costs.

Due to the missing link between models and documents the monitoring and securing of the information's consistency requires permanent resources.

3.4.2 Multiple Information Sources

The examined publications mention the implied risks of multiple information sources. Several sources create confusion about the valid one. At AVL several similar or complementing working products (SysML diagrams, FTAs, Microsoft Excel sheets) of the element safety mechanisms exist. Taking the consistency problems into account, implies a potential risk for the correct creation of functional safety requirements. Depending on the domain specific background of the developer, several sources might be used from different developers simultaneously, which might result in the previous mentioned problems of consistency.

3.4.3 Ambiguous Requirements

According to the examined publications, the advantage of text-based requirements is their comprehensibility for laypersons. The disadvantage is the lack in clarity. Even though the functional safety requirements created by AVL minimize this problem by using templates, their use is still open to interpretation and hence, possible faults.

To illustrate this problem, the following Figures 3.7 and 3.8 show a functional safety requirement for the EDRV and the OBC. Both functional safety requirements are formally correct and the intended function of both requirements is the same. Nevertheless, two different sentence templates are used for their description, which can lead to misunderstandings in later development processes.

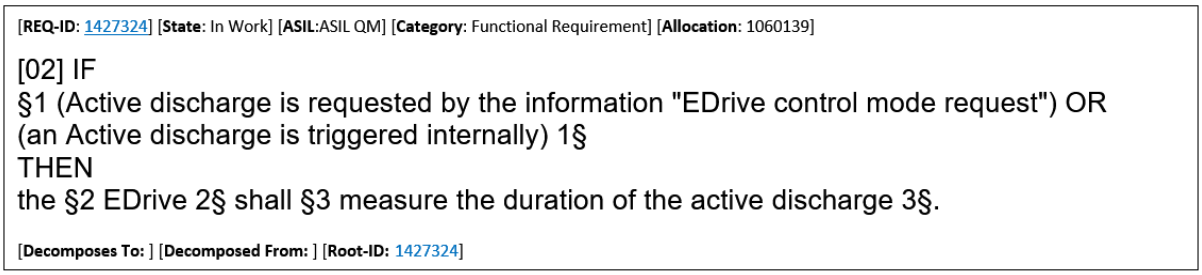

Figure 3.7: Example 1 of unambiguous problems [4]

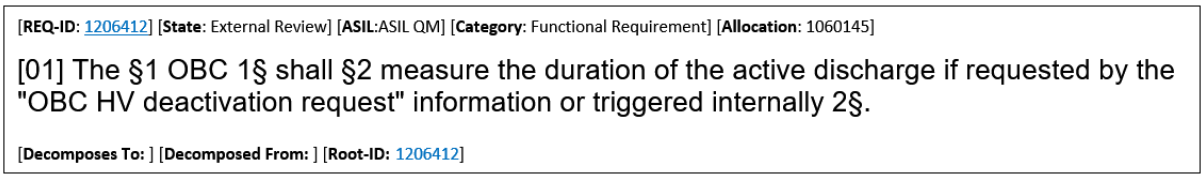

Figure 3.8: Example 2 of unambiguous problems [4]

3.4.4 Risk of Reuse

Another topic which is taken up by several publications is the reuse of already developed parts of a system. At first glance text-based requirements are ideal for this task and the AVL developers seize this possibility, but the use of existing requirements for different parts of a system requires many conscientious adaptations of the requirements. This is shown in the following example.

The Figures 3.9 and 3.10 show almost the same functional safety requirements. The requirements detect an “Engine failure” (red) and trigger two different DMs (blue). The developer reused the requirement of Figure 3.9 to create the functional safety requirement of Figure 3.10 and did accidentally not adapt the whole requirement. The correct adaptation is shown in Figure 3.11 (green). In the correct functional safety requirement, the “EDRV failure” should trigger the specific DM instead of the “Engine failure”. The consequence of the faulty reuse of a functional safety requirement is a probably undetected “EDRV failure” and an “Engine failure” which triggers the DM for the EDRV torque limitation (DM[HCU].06).

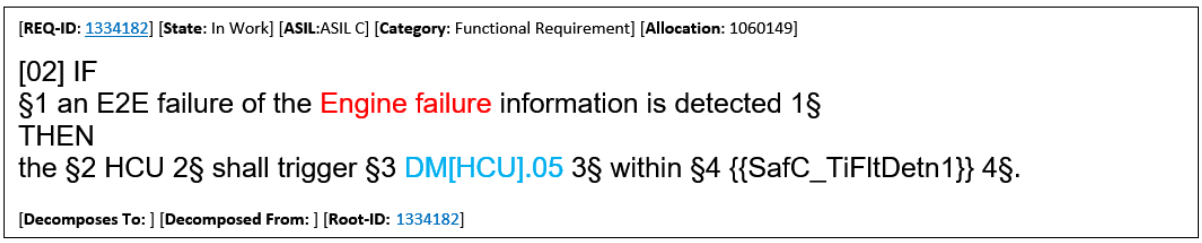

Figure 3.9: Example 1 for the risk of reuse, derived from [4]

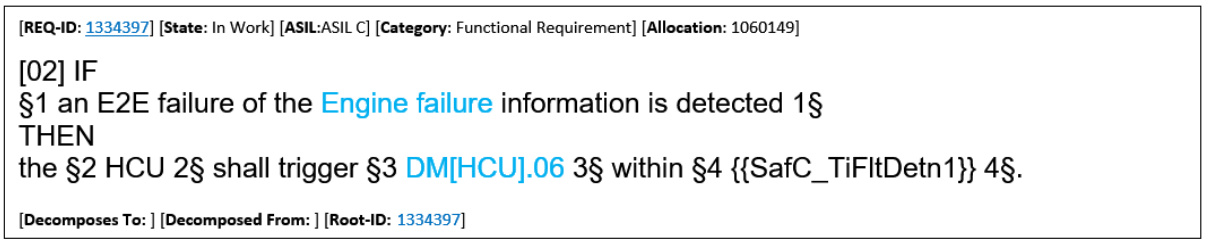

Figure 3.10: Example 2 for the risk of reuse, derived from [4]

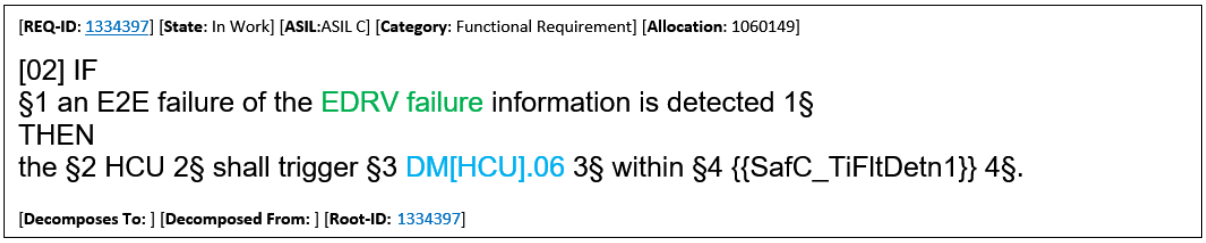

Figure 3.11: Example 3 for the risk of reuse, based on [4]

3.4.5 Limitation of Natural Language

Another weak spot of natural language is the illustrative depiction of exact matters. Simple issues create voluminous text-based requirements. These text-based requirements are cumbersome and are therefore error-prone and easily misinterpreted.

An example is shown in Figure 3.12. The functional safety requirement is also based on a type two requirement pattern template, such as the previously shown figures, but the description is far more complicated.

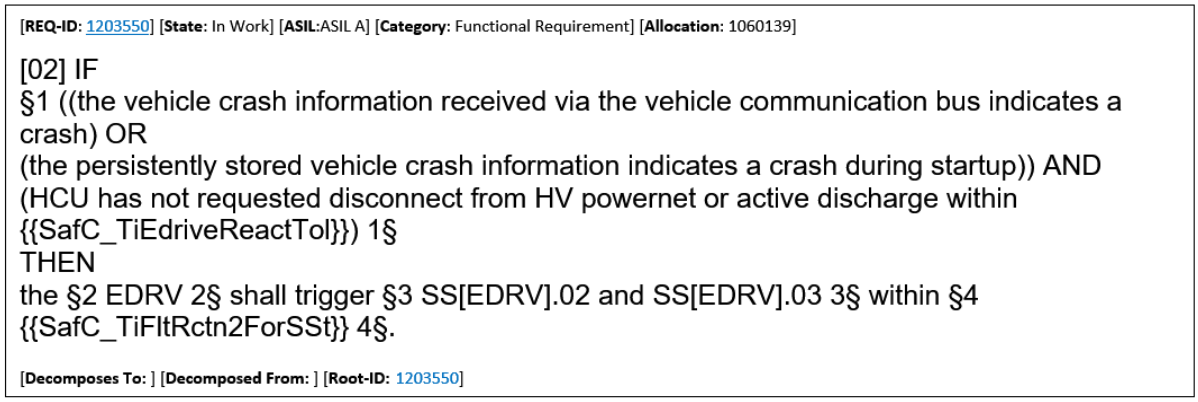

Figure 3.12: Example for depiction weakness of natural language [4]

3.5 Demands on the Model-Based Solution

The description of functional safety requirements in the functional safety concept of AVL involves several problems. The developed solution of this thesis shall improve the development process with a

model-based approach with SysML in Integrity Modeler and Matlab/Simulink. Hence, the developed model-based approach shall pay attention to the following requirements:

- improve the connectivity to other working products
- allow text-based requirements to be derived from modeled requirements
- create unambiguous functional safety requirement
- allow the reuse of functional safety requirement artifacts and demand exact definitions
- create easily interpretable elements

Due to the limited scope of the thesis, described in Section 1.3, only the foundation for these demands can be laid.

4 METHODS AND CONCEPT DEVELOPMENT

As mentioned in Chapter 1, the functional safety concept serves as the information basis of the thesis. After the previous section in which the current situation and development foundation is explained, the following chapter focuses on the methodology and the concept development. The aim was to extract relevant information from the functional safety concept by analyzing it and finding patterns in it, which can then lead to the development of a library of reusable blocks. This library should then undergo a proof of concept. The following sections reflect both the methodology of the analysis and the step-by-step comprehension of the functional safety concept, which led to a solution.

In Section 4.1 the analysis which led to the development of a concept will be described. Section 4.2 shows the realizations regarding the communication signals between elements and within elements. Although these realizations took place simultaneously to the described content of Section 4.1, it will be examined separately to improve the readers overall comprehension.

In the Sections 4.3 to 4.6 the method to transfer the developed concept of Section 4.1 to a solution will be described.

In Chapter 5 the findings of this solution will be presented.

4.1 Analysis of the Functional Safety Concept

For an overall view it was decided to examine the whole functional safety concept. This included all safety goals, the safety mechanisms, FTAs, GSN diagrams etc.. The whole content of the functional safety concept is described in Section 3.3.

It was concluded that an efficient analysis for patterns within this extensive document requires a fragmentation in parts, based on the relevance of the content of these parts and a reduction of the scope. The reduction of scope is described in Section 4.3.

Due to the structure of the functional safety concept, where for example all FTAs of the safety goals are gathered in one section of the document and all safety mechanisms of the safety goals are gathered in another section of the document, the first task was to identify relevant document parts.

Criteria for relevant document parts were that their content can be influenced within the AVL development process with a model-based approach and were not already based on a model-based approach, or the parts provide helpful information for the understanding of other document parts.

The “Functional Safety Requirements” section, described in Section 3.3.4 of this thesis contains information about mechanisms to prevent safety goal violations. This part of the document was

identified to be mainly text-based and potentially improved with a model-based development process within the AVL and therefore further examined.

The requirement sentence templates of the “Introduction” section and the “Safety goals” were identified to provide helpful information for the understanding of functional safety requirements. Their description can be found in Section 3.3.1 and 3.3.3, respectively.

All FTAs and GSN diagrams were of secondary relevance for this thesis and not further examined, because they did not fulfill the defined criteria.

Within the “Functional Safety Requirements” section the functional safety requirements were identified as the most relevant parts of the functional safety concept, because they are mainly text-based. They are structured in different levels. On powertrain level in powertrain safety mechanisms (PTSM) and on element level in element safety mechanisms (ESM, SRF, SS, DM) and contain key information to fulfill the safety goals. As explained in Section 3.3.4, these functional safety requirements give detailed information about how to prevent safety goal violations and are based on requirement sentence templates.

4.1.1 Functional Safety Requirements on Powertrain Level

On the powertrain level the functional safety requirements were identified to describe the general safety mechanisms, so called powertrain safety mechanisms, such as the avoidance of the high voltage incident of SG07, or the avoidance of the HV battery outgassing of SG08. Due to the general nature of these requirements, they were identified to differ too much to be modeled with blocks.

Furthermore, the communication organization between the elements involved in a safety goal, e.g. HCU and HVBATT in SG07 are visible on powertrain level. This is described by activity diagrams for every specific safety goal. The functionality of the communication between different elements, to satisfy safety goals, was identified as first recurring pattern.

In Figure 4.1 one can see a part of the powertrain safety mechanism of SG07, represented by a SysML activity diagram. The communication structure between the ESP (Electronic Stability Control), PEPS (Passive Entry Passive Start) and HCU, the connection of ESMs and SRFs, and the internal generated signals of the ESP and the PEPS are visible[11].

Although the realizations regarding communication signals of the elements took place simultaneously to the following realizations, they will be examined separately in Section 4.2. This is done to improve the understanding of Section 4.

[11] It is important to mention that the functional safety concept, provided by AVL, contained outdated diagrams of an earlier development stage, which only partly matched the textual information and as a result, those figures were only a guideline and might lead to contradictions with other figures of this thesis.

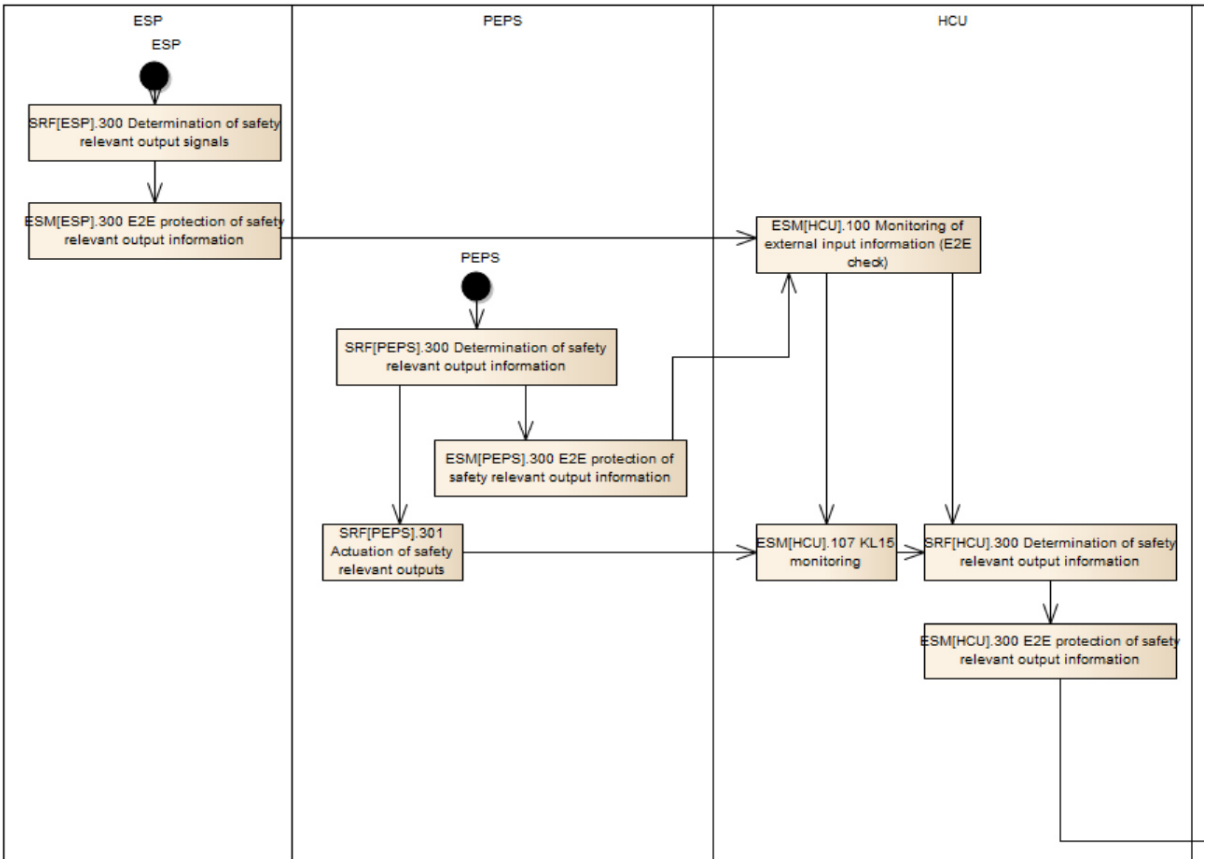

Figure 4.1: Part of the powertrain safety mechanism of SG07 [4]

4.1.2 Functional Safety Requirements on Element Level

After the powertrain level, the functional safety requirements of the element level were examined. On element level are the functional safety requirements gathered for every element (e.g. ESP, HCU, HVBATT etc.) in the ESMs, SRFs, SSs and DMs of the specific element.

The first important realization on this level were the ESMs, SRFs, SSs and DMs of the elements being potentially used to satisfy several safety goals. This means that one ESMs, SRFs, SSs and DMs is probably used multiple times for different safety goals. Hence, the functional safety requirements as part of the ESMs, SRFs, SSs and DMs are also used to satisfy one or several safety goals.

Figure 4.2 shows the information exchange of ESMs and SRFs of the ESP in an Internal Block Diagram. Such diagrams are part of the description of every element in the functional safety concept. It was identified as the model-based equivalent of the text-based safety mechanisms (e.g. ESM, SRF, SS and DM) and provides an overview of all necessary ESMs and SRFs to prevent safety goal violations of one or several safety goals, related to functionalities of the ESP. As one can see, the “ESM[ESP].300 E2E protection of safety relevant output information” is a block, which is used by several signals, like e.g. “Gear shift forbidden” and “Vehicle speed” information. These signals are used for different safety

goals. Furthermore, the communication structure which was shown on powertrain level for one PTSM in Figure 4.1 is visible. Due to readability reasons are SSs and DMs not visible in Figure 4.2.

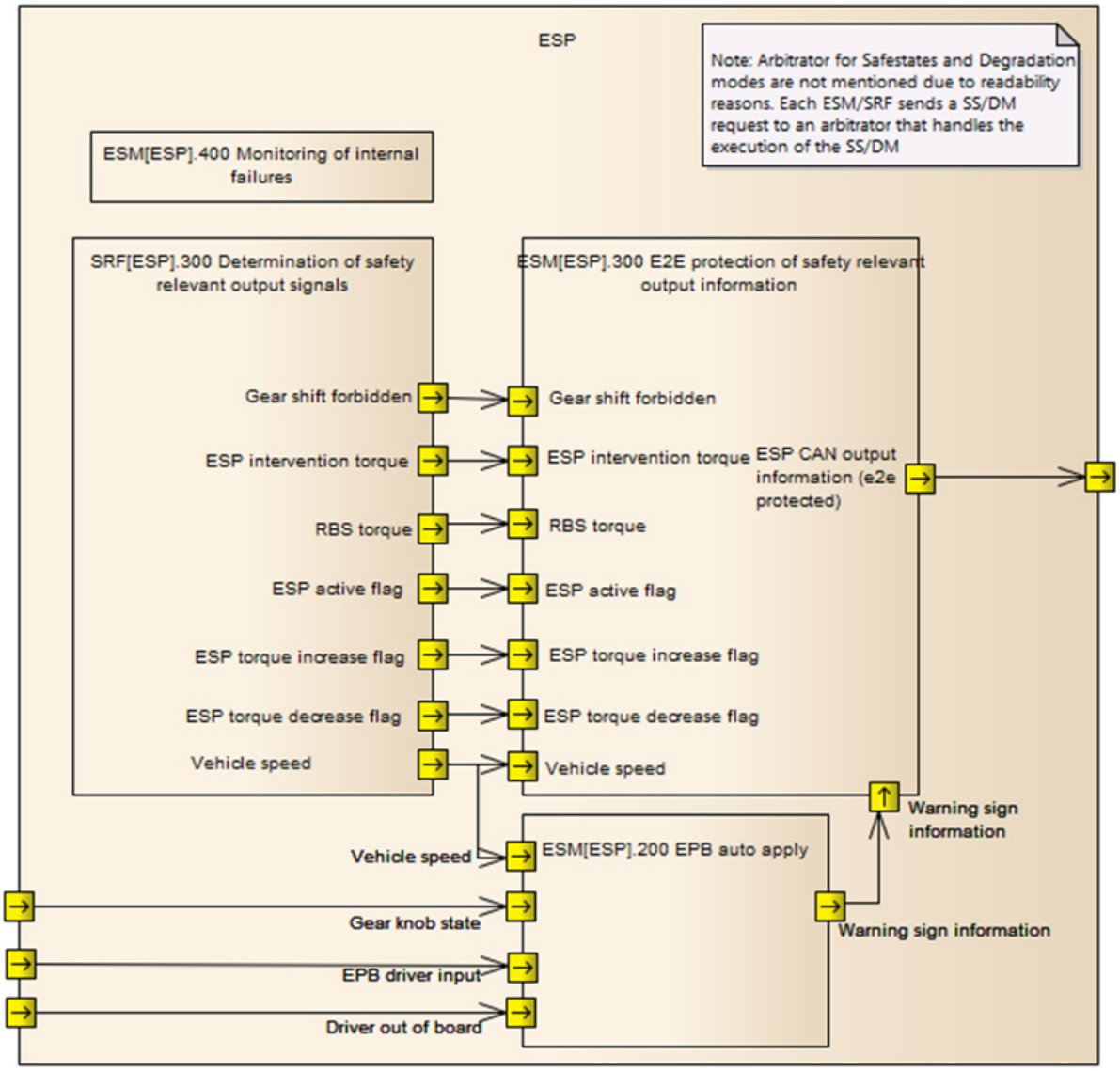

Figure 4.2: Internal block diagram of the ESP [4]

The second important realization on element level were the more specific functional safety requirements. In contrary to the general nature of the requirements on powertrain level, the requirements on element level contain clear similarities between each other.

The similarities of the functional safety requirements on element level were identified as the most relevant parts for a further analysis of the functional safety concept for patterns. The requirement pattern templates and the keywords and semantic provided very important additional information.

Figure 4.3 shows three exemplary functional safety requirements which are based on the same requirement sentence templates. The content of these requirements is almost identical. Every requirement demands to check the “vehicle crash information”, the “persistently stored vehicle crash

information", and an HCU request and trigger a specific safe state. The requirements differ mainly, because the requirements are used for different elements (EDRV, DCDC, OBC).

Figure 4.3: Exemplary similarities of functional safety requirement on element level based on a template [4]

Examples like these led to the realization that the functional safety requirements on element level are the promising parts of the functional safety concept, to be modeled with predefined and reusable blocks from a library. As a result, would text-based requirements and the requirement sentence templates on this level be obsolete.

The following Figure 4.4 displays the previous realizations with simple entity-relationships. Entity-relationships are not explained in this thesis, but can be found in [36].

As described are the functional safety requirements based on requirement sentence templates. The requirements are used on powertrain level for PTSMs and on element level for ESMs, SRFs, SSs and DMs. The requirements for the PTSMs differ too much, but on element level are clear similarities

between different functional safety requirements. The part of the figure colored in blue shows the approach for the implementation of blocks and a block library. The block library offers different blocks, which can be used to model functional safety requirements on element level. The requirement sentence templates are then dispensable for these requirements (dashed connection), because the functional safety requirements are model-based (green).

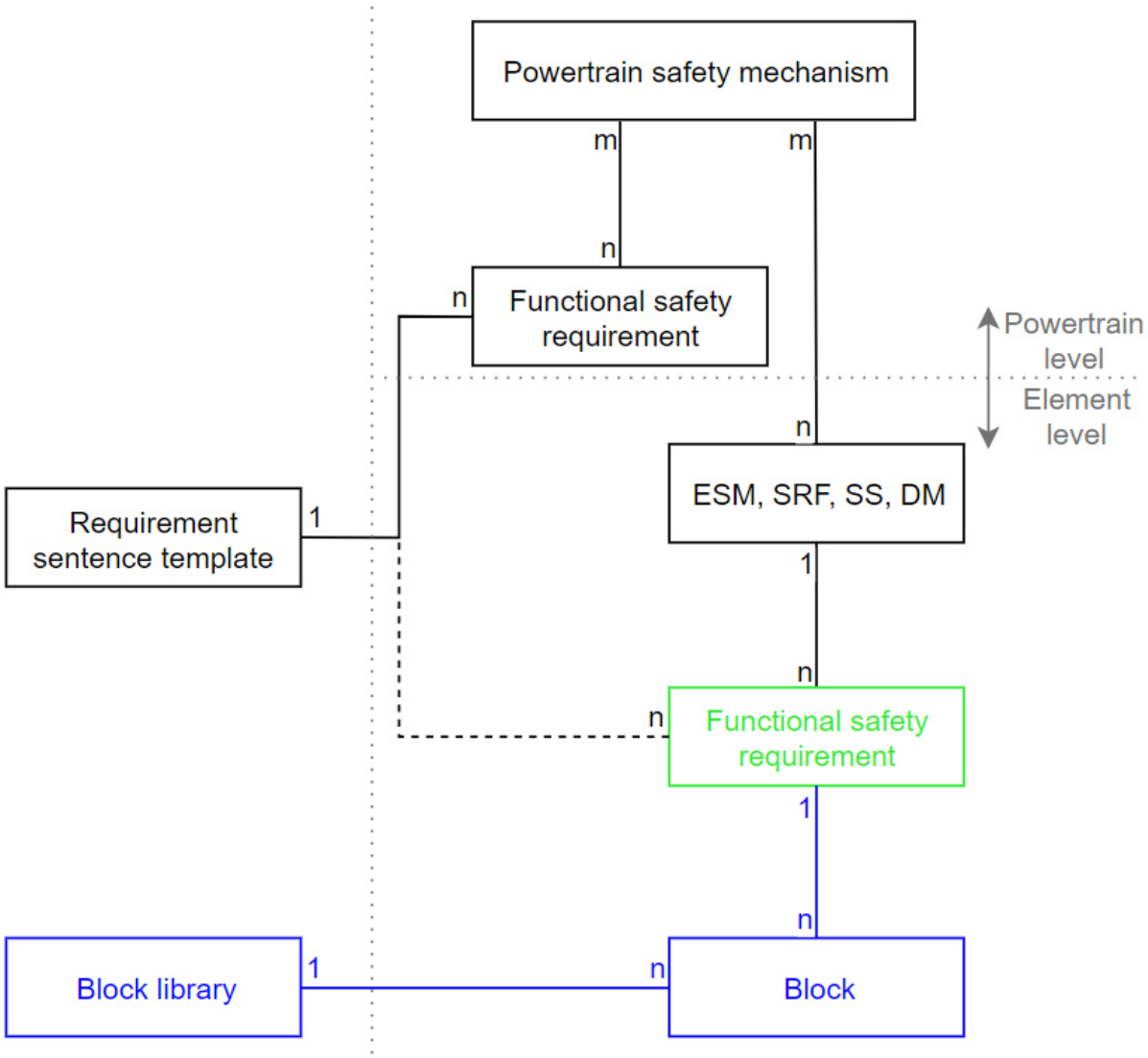

Figure 4.4: Implementation of the block library and blocks

4.2 Communication Signals

Beside the realizations described in Section 4.1, further realizations were made, regarding the communication signals between elements and within elements. To improve the readers understanding, these realizations are separated from Section 4.1 and presented here.

4.2.1 Signal Types for Communication

The communication within every element in the ESMs, SRFs, SSs and DMs and between elements were identified to communicate with Boolean signals (“True”, “False”), or with a range of values and qualifier values. The Boolean signals were imagined as a simple 1 or 0 information or trigger, respectively. In contrary to this, there are signals such as e.g. the vehicle speed, which provides a range of values. Furthermore, qualifier values exist, which provide information of e.g. a specific status or mode of a system. All these communication signals were identified to be used in the functional safety requirements in different combinations.

The following figures shows two exemplary functional safety requirements. The first requirement in Figure 4.5 shows an example for a specific state, which can be requested and requires the transmission of a trigger signal to a safe state within a specific time.

The second functional safety requirement, displayed in Figure 4.6, necessitates checking several different states and whether the range of “vehicle speed” signals are below a specific value.

[**REQ-ID**: 1427320] [**State**: In Work] [**ASIL**:ASIL A] [**Category**: Functional Requirement] [**Allocation**: 1060139]

[02] IF
§1 (Disconnect from HV powernet is requested by the information "EDrive control mode request") 1§
THEN
the §2 EDrive 2§ shall trigger §3 SS[EDRV].02 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1427320]

Figure 4.5: Example of a requirement to check for a specific state [4]

[**REQ-ID**: 1426035] [**State**: In Work] [**ASIL**:ASIL A] [**Category**: Functional Requirement] [**Allocation**: 1060149]

[02] IF
§1 ((the KL15 is LOW) OR
(the OBC state information received via vehicle communication bus indicates that charging is completed)) AND
(vehicle speed is below {{Power_Down_Vehicle_Speed_Threshold}}) 1§
THEN
the §2 HCU 2§ shall trigger §3 SS[HCU].08 3§ within §4 {{SafC_TiFltRctn2ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1426035]

Figure 4.6: Example of a requirement to check the signal range [4]

4.2.2 Connection Types for Communication

For the communication between elements, a distinction was identified between hard-wired communication and communication by using a CAN-Bus.

Unlike hard-wired electrical lines like the HVIL (High Voltage Interlock Loop), or the KL15(HW) (Clamp 15 (Hardwired)), the communication via CAN-Bus is E2E protected and specific ESMs

are used to process the signals on the outgoing and the receiving port. This can also be seen in Figure 4.1.

The E2E protection of CAN-Bus signals is also a specific recurring pattern, which will be shown exemplarily in Section 5.2.4.

4.3 Isolation of SG07 Specific Requirements

After the identification of the text-based functional safety requirements of the ESMs, SRFs, SSs and DMs as potentially describable with a model-based approach of recurring library blocks and the ascertainment regarding the communication signals, the observation horizon was reduced to only one safety goal. The reduction of scope allows an easier search for patterns.

On the one hand, the potential safety goal for examination must be extensive enough to get a cross-section of the functional safety concept, but on the other hand it should still be comprehensible. Hence, the "SG07 – Avoid unintended HV" was chosen, because it fulfills these demands.

The purpose of this safety goal is the protection of people from an HV incident. This could happen in the case of an HVIL interruption, a crash of the vehicle, or when the insulation resistance is too low. Furthermore, a discharge of the system is also triggered at each shut down of the system.

The HVIL is a hard-wired loop which passes through several HV elements, like OBC and EDRV and the interruption of the HVIL means that these elements might be accessible and therefore a potential source for HV incidents.

Appropriate functional safety requirements describe SG07 on the powertrain level. The according figure is shown in Section 6 (Figure 6.2), where an extensive example of this safety goal and especially the HVIL will be presented.

As explained in detail in Section 3.2, the elements and their SRFs, ESMs, etc. are part of a model, including all safety goals and therefore the functional safety requirements can be confusing to retrace. To prevent such confusion, the diagram of the powertrain safety mechanism, as shown in Figure 4.1, was identified to give an overview of the specific safety goal. It was used in combination with the Internal Block Diagram of each element (e.g. ESP, HCU, PEPS, EDRV, etc.) and although these depictions were already outdated, it helped to differentiate between relevant und irrelevant ESMs and SRFs describing the SG07.

Another important remark which was mentioned in Section 4.1 needs to be reasserted. Every SRF, ESM, SS and DM may contain several functional safety requirements, which are used for one or several safety goals. Hence, it was also necessary to identify the relevant requirements of SG07.

At the end of this step, all relevant ESMs, SRFs, SSs and DMs and their relevant functional safety requirements were gathered.

4.4 Grouping Similar Requirements

After the identification of relevant information for SG07, the functional safety requirements were regrouped, regardless of the elements they were used in. Only the requirement pattern templates and textual similarities were relevant qualities for the comparison.

In this section several examples for regrouped requirements will be presented.

One example was already presented in Figure 4.3. Those three functional safety requirements' content is almost identical. Every requirement demands to check the same conditions. The requirements differ mainly, because of their use for different elements (EDRV, DCDC, OBC).

Another example for almost identical functional safety requirements is shown in Figure 4.7. In this example the requirements demand to apply E2E protection on safety relevant output information. Their similarity was the reason to be grouped.

[REQ-ID: 1165385] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060141]

[01] The §1 HV Battery 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForHvbat}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1165385]

[REQ-ID: 1165553] [State: External Review] [ASIL:ASIL B] [Category: Functional Requirement] [Allocation: 1060143]

[01] The §1 DCDC 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForDcdc}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1165553]

[REQ-ID: 1165471] [State: External Review] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation: 1060145]

[01] The §1 OBC 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForObc}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1165471]

[REQ-ID: 1146163] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060149]

[01] The §1 HCU 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForHcu}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1146163]

[REQ-ID: 1146120] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060139]

[01] The §1 EDrive 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForEm}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1146120]

Figure 4.7: Example 2 for grouping similar requirements [4]

The previous two examples showed almost identical requirements, but on closer examination also functional safety requirements with more differences were found, where grouping made sense. In Figure 4.8 all four functional safety requirements demand a check for an E2E failure. The main

difference is the requested reaction. Requirement one to three are based on the same requirement sentence template and trigger a safe state. In contrary is the fourth requirement another template and demands to set a specific fault.

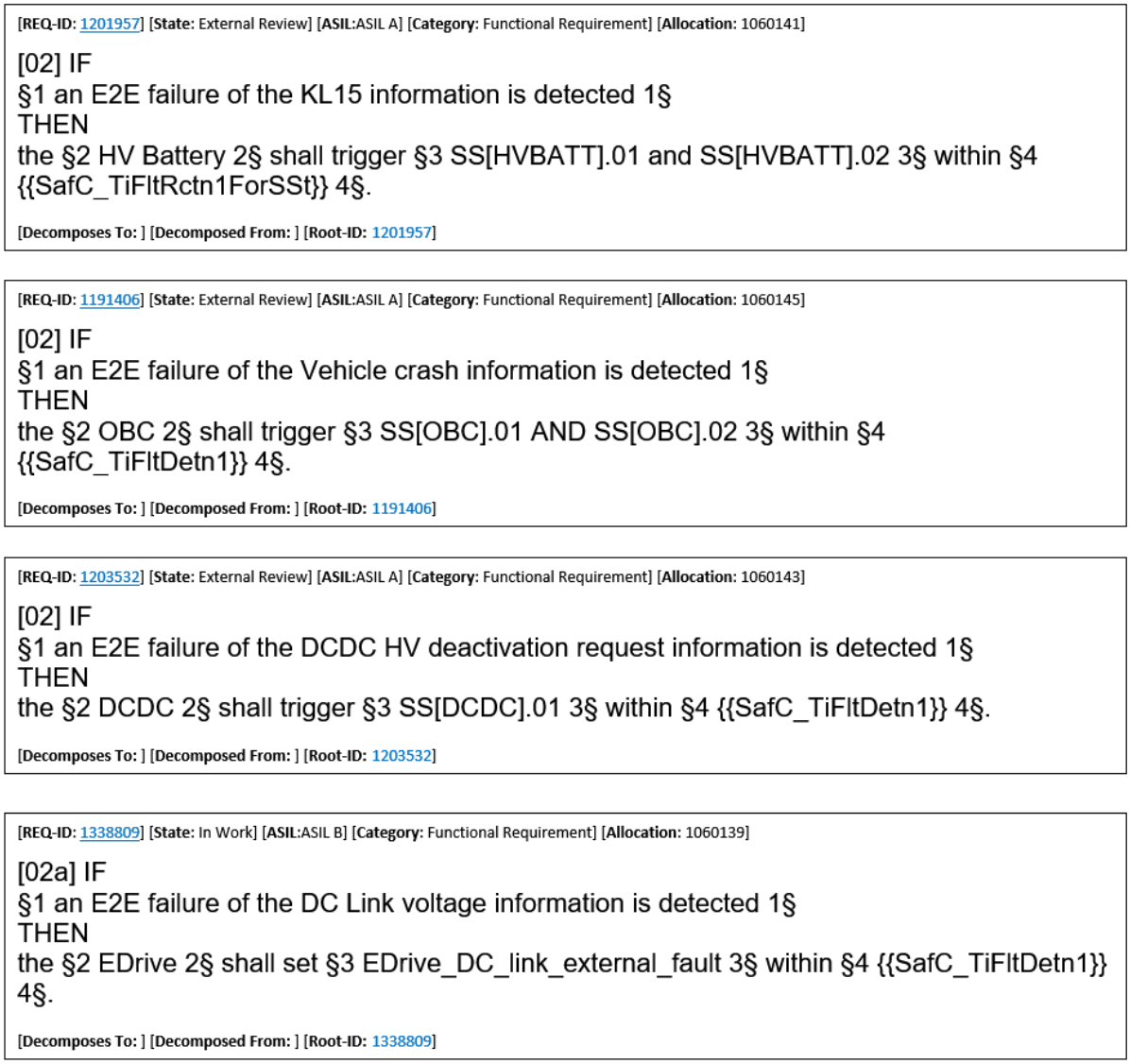

Figure 4.8: Example 3 for grouping similar requirements [4]

The fourth example for grouping, shown in Figure 4.9, differs even more than the previous examples. The only common part of these functional safety requirements is the demand to “remember the occurrence of this event by increasing a persistently stored counter”. This was decisive for the grouping of these requirements.

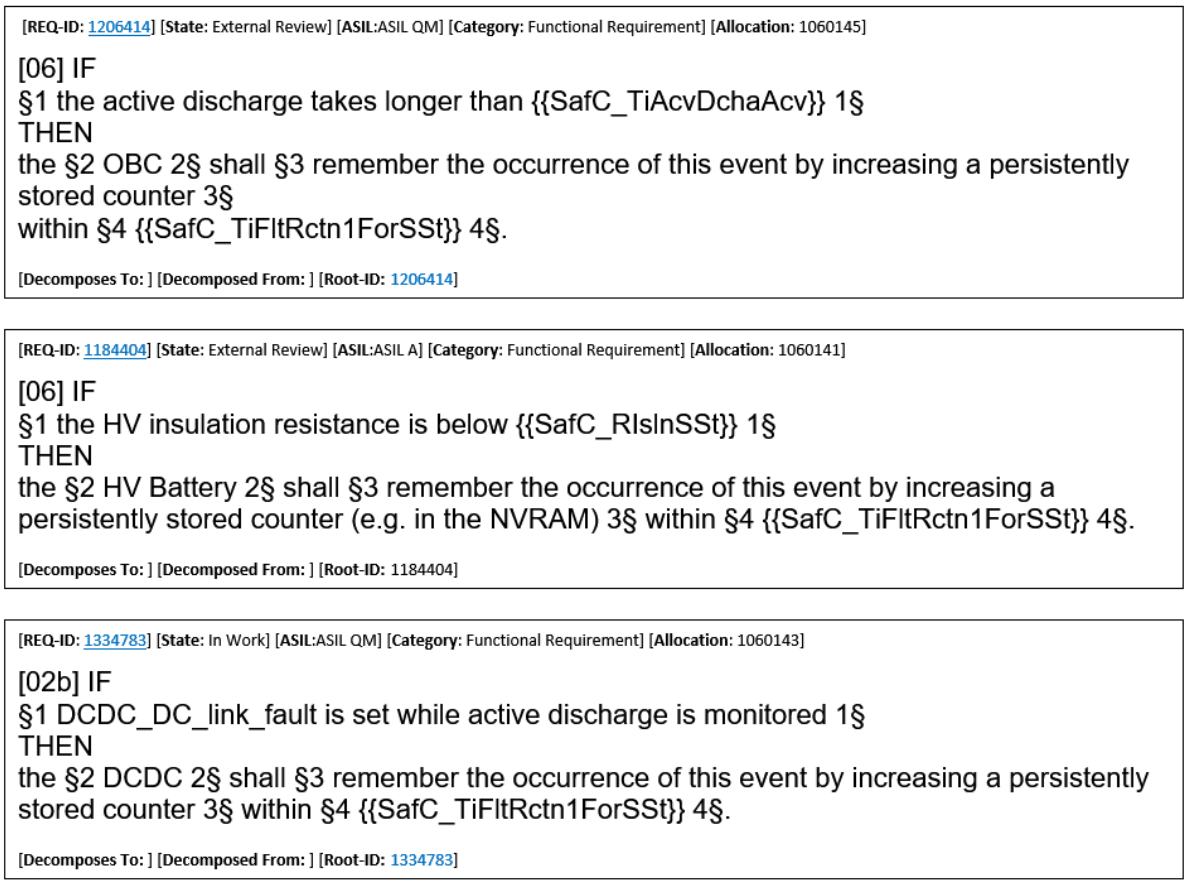

Figure 4.9: Example 4 for grouping similar requirements [4]

Other functional safety requirements were grouped, similar to presented examples.

4.5 Development of a Block Library with SG07

Once the functional safety requirements were regrouped, every requirement was separated in different functional parts. The possibility to separate every requirement in parts is a major finding and will therefore be shown in Section 5.1. Moreover, signal words and demands for periodicity and duration were identified as important information in every functional safety requirement and are therefore also shown in Section 5.1.

Subsequently, Matlab and Simulink were used to create a model from these requirements. Therefore, blocks were developed or used from the existing Simulink library. These blocks reflect the text-based requirements in a Simulink model. The developed blocks were described with basic code in Matlab and by predefined Simulink blocks. If necessary an input mask was also created. The developed blocks are described in Section 5.2 of the thesis. The mask as a graphical user interface (GUI) provides more details about the functionality of a block and input possibilities. The input possibilities were identified to be necessary to specify the block according to the functional safety requirement it should portray. Once most of the functional safety requirements were modeled, recurring blocks were added to a

custom library. The method of creating custom blocks and a custom library in Simulink will not be described in this thesis, but a description can be found at [33].

Systems and subsystems were created in Simulink following the system architecture and the structure of the Internal Block Diagrams of the elements in the functional safety concept. These systems were filled with ESMs, SRFs, SSs and DMs relevant for SG07. Afterwards the blocks in the custom library were used to remodel the requirements of every ESM, SRF, SS and DM. Finally, the blocks were connected with each other according to the demands of SG07.

4.5.1 Proof of Concept with SG08 & SG09

After SG07 was modeled, the same custom library was used for modeling SG08 and SG09, thus proofing the concept and the usability of the library. Therefore, relevant functional safety requirements were again isolated and systems and subsystems were created, filled with blocks and connected with each other.

Although the safety goals “SG08: Avoid HV battery outgassing” and “SG09: Avoid sudden loss of LV system supply” are not similar to SG07 at first glance, the library was applicable to those safety goals and no additional blocks were necessary.

4.6 Transfer of the Model to SysML in PTC Integrity Modeler

To create a model in SysML with Integrity Modeler, AVL provided a sandbox model including the system levels 0 to 3, which is closely related to the P2-HEV functional safety concept. To match this specific functional safety concept, some adaptations to the provided model were applied, such as additional “Activities”.

Subsequently, a library of “Call Behavior Action”-blocks and pins[12], similar to the Simulink library was developed. The model was reproduced with the Call Behavior Action-blocks in the Activity Diagrams of every element (ESP, HCU etc.)[13].

As the functionality of Integrity Modeler and Matlab/Simulink differ, some adaptations were necessary and parts of the Simulink functionality were not implementable. These differences will be discussed in Section 5.4.

[12] Pins are used in SysML as the equivalent to ports in Simulink

[13] AVL replaced for functional safety developments the SysML modeling software and the used diagrams. As a result, there is a discrepancy between the Call Behavior Action-blocks used in Activity Diagrams (new development process) and the Internal Block Diagrams presented in e.g. Figure 3.4 and Figure 4.2 (old development process). Because of the outdated functional safety concept all SysML diagrams of Chapter 3 and 4 are outdated. The created models of this thesis are Activity Diagrams in Integrity Modeler.

5 FINDINGS

After focusing on the method and the concept development in Section 4, this section deals with the findings and the simplifications made. In the first part the findings within the functional safety requirements will be described. Afterwards the developed library and blocks will be shown. To stress the usage potential of the library, some statistical results will be shown in Section 5.2.6. Finally, some simplifications and differences in the tool usage and the specific realization in each tool will be explained. To avoid repetition, the focus will be especially on Simulink. Only if appropriate or necessary will the equivalent solution also be presented in Integrity Modeler.

5.1 The Functional Safety Requirements

For the search for patterns the observation horizon was reduced to SG07. To satisfy SG07 approximately 160 functional safety requirements are used. Only 27 predefined blocks were necessary to model most of these requirements. An overview of all blocks and the specifics of the blocks will be shown in Section 5.2.

The major patterns in the functional safety requirements will be explained exemplarily with the requirements presented in Figures 5.1 to 5.3. These requirements were chosen for the description of the findings, because they are comprehensible and pithy.

5.1.1 Separation of Requirements in Parts

The first finding in the functional safety requirements is the pattern that at least the first textual part of the functional safety requirement can be separated from the second part. For more complex requirements a partitioning in even more parts is possible. The semantics “and” and “or” were indicators for the separation.

This finding allowed the focus of the investigation on the specific requirement parts, without considering the rest of the requirement and was the first step to standardized blocks. Three examples are shown in Figure 5.1. The possibilities for partitioning are highlighted. The first parts in the requirements, which contains the examination, are colored red, whereas the second parts with the reaction are colored blue. For the second and the third requirement it is possible to split the examination part of the requirement in three separate parts. Furthermore, the third requirement’s reaction part can be separated in two parts.

[**REQ-ID:** 1427322] [**State:** In Work] [**ASIL:**ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060139]

[02] IF
§1 (Active discharge is requested by the information "EDrive control mode request") 1§
THEN
the §2 EDrive 2§ shall trigger §3 SS[EDRV].03 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1427322]

[**REQ-ID:** 1206355] [**State:** In Work] [**ASIL:**ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060143]

[02] IF
§1 ((the vehicle crash information received via the vehicle communication bus indicates a crash)
OR
(the persistently stored vehicle crash information indicates a crash during startup)) AND
(HCU has not requested disconnect from HV powernet or active discharge within
{{SafC_TiEdriveReactTol}}) 1§
THEN
the §2 DCDC 2§ shall trigger §3 SS[DCDC].01 3§ within §4 {{SafC_TiFltRctn2ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1206355]

[**REQ-ID:** 1203550] [**State:** In Work] [**ASIL:**ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060139]

[02] IF
§1 ((the vehicle crash information received via the vehicle communication bus indicates a crash)
OR
(the persistently stored vehicle crash information indicates a crash during startup)) AND
(HCU has not requested disconnect from HV powernet or active discharge within
{{SafC_TiEdriveReactTol}}) 1§
THEN
the §2 EDRV 2§ shall trigger §3 SS[EDRV].02 and SS[EDRV].03 3§ within §4
{{SafC_TiFltRctn2ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1203550]

Figure 5.1: Example for partitioning of requirements, derived from [4]

5.1.2 Signal Words for Block Derivation

The second finding within the functional safety requirements are signal words and phrases. In the analysis of the requirements were signal words or phrases, respectively, recognized to indicate the necessity of a specific function, which the requirement should fulfill. These functions were identified to be potentially modeled with blocks and a series of connected blocks describe a whole functional safety requirement. For the fulfillment of the functions with blocks, it was assumed that the connection for the information exchange is based von block ports.

Typical signal words and phrases and the derived functionality of a corresponding block are shown in Table 5.1.

Table 5.1: Examples of signal words and the derived block functionality

Signal word	Derived block functionality
Indicate	Compare input against a specific value for equality
Not equal	Compare input against a specific value for inequality
Exceed	Compare input for its size
Below	Compare input for its size
Perform an end-to-end check	Check the input for the E2E status
Apply end-to-end protection	Carry out E2E protection to the output
Remember the occurrence of this event	Keep the information about an event in a memory
Determine	Get input from an element
Measure duration	Determine the duration between two events

Figure 5.2 shows six examples for functional safety requirement which contain typical signal words or phrases (red and blue).

The first three requirements are based on a type two requirement sentence template and shall trigger a signal for a safe state within a specific time. But they differ by demanding to check for a specific value (first requirement), for inequality (second requirement) and the size (third requirement), respectively. The third and the fourth requirement are based on different requirement pattern templates and therefore trigger different actions, nevertheless does the fourth requirement also demand a check for size. Moreover, the fourth requirement demands to remember the comparison of size (blue). The fifth and the sixth requirements demand to perform or apply specific actions, respectively.

The developed blocks derived from these findings are described in Section 5.2. The reuse of such blocks can be realized with a library.

[REQ-ID: 1426029] [State: In Work] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation: 1060149]

[02] IF
§1 the vehicle crash information received via the vehicle communication bus indicates a crash 1§
THEN
the §2 HCU§ shall trigger §3 SS[HCU].08 3§ within §4 {{SafC_TiFltRctn2ForSSt]}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1426029]

[REQ-ID: 1142215] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060139]

[02] IF
§1 the EDrive control mode request is not equal with the actual EDrive control mode 1§
THEN
the §2 EDrive 2§ shall trigger §3 SS[EDRV].01 §3 within §4 {{SafC_TiFltRctn1ForSSt}} 4§

[Decomposes To:] [Decomposed From:] [Root-ID: 1142215]

[REQ-ID: 1183200] [State: External Review] [ASIL:ASIL QM] [Category: Functional Requirement] [Allocation: 1060139]

[02] IF
§1 the active discharge time failure counter exceeds {{SafC_CtrThdForAcvDchaTrigSSt}} 1§
THEN
the §2 EDrive 2§ shall trigger §3 SS[EDRV].04 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§

[Decomposes To:] [Decomposed From:] [Root-ID: 1183200]

[REQ-ID: 1184404] [State: External Review] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation: 1060141]

[06] IF
§1 the HV insulation resistance is below {{SafC_RIsInSSt}} 1§
THEN
the §2 HV Battery 2§ shall §3 remember the occurrence of this event by increasing a persistently stored counter (e.g. in the NVRAM) 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1184404]

[REQ-ID: 1147337] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060149]

[01] The §1 HCU 1§ shall §2 perform an end-to-end check on all safety relevant input communication signals 2§ periodically each §4 {{SafC_TiDetn1CanInpInfo}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1147337]

[REQ-ID: 1146163] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060149]

[01] The §1 HCU 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForHcu}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1146163]

Figure 5.2: Exemplary functional safety requirements demanding functions, derived from [4]

5.1.3 Demands for Periodicity and Duration

Another finding that was made in the functional safety requirements is that every requirement gives information about either the periodicity a requirement should be fulfilled, or the duration for a requirement to be executed, or both. This information about the time is usually defined in curly brackets at the end of each requirement. This is another recurring pattern in the functional safety requirements. The implementation of this information in blocks is solved by input masks. This will also be shown in Section 5.2.

The following figure displays two examples of functional safety requirements. The first requirement demands a periodical check (red) and the second requirement defines the demand to fulfill an action within a specific period (red).

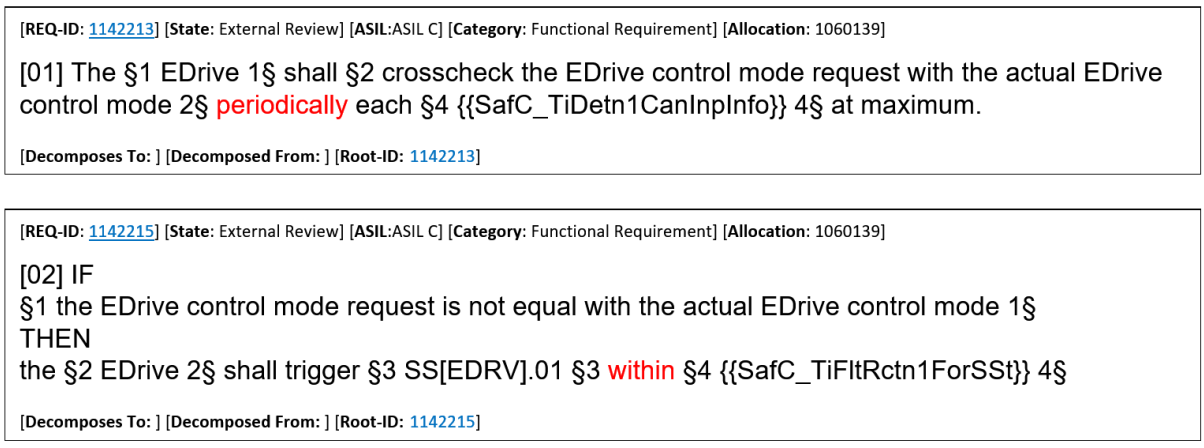

Figure 5.3: Periodicity and duration of signals, derived from [4]

Beside periodicity and duration, curly brackets are also used in functional safety requirements to define other parameters.

5.2 The Block Library

For SG07, SG08 and SG09 it is possible to model most investigated functional safety requirements and hence, satisfy all examined safety goals with 27 predefined blocks. These blocks are predefined in their functionality and, with a few exceptions, in the number of input and output ports. To model a functional safety requirement, the blocks can simply be dragged from the library into the model and then be connected.

The Figures 5.4 and 5.5 show a part of the library in Simulink and in Integrity Modeler.

It is important to mention that the Library in Simulink offer a directory tree in the left and a preview window in the right part of the figure. The library of the Integrity Modeler also offers a directory tree but the visible blocks on the right part of Figure 5.5 only serve as help for the reader of this thesis to

imagine how the library blocks in Integrity Modeler can look like in use. The differences of the libraries will be discussed in more detail in Section 5.4.

Because of the wide variety of predefined blocks in Simulink, some of the blocks can be used unrevised, or in a subsystem as part of the developed blocks. Due to the concept of SysML as a modeling language all the blocks in Integrity Modeler are individual blocks, but nevertheless reflect the same intended functionality as in Simulink.

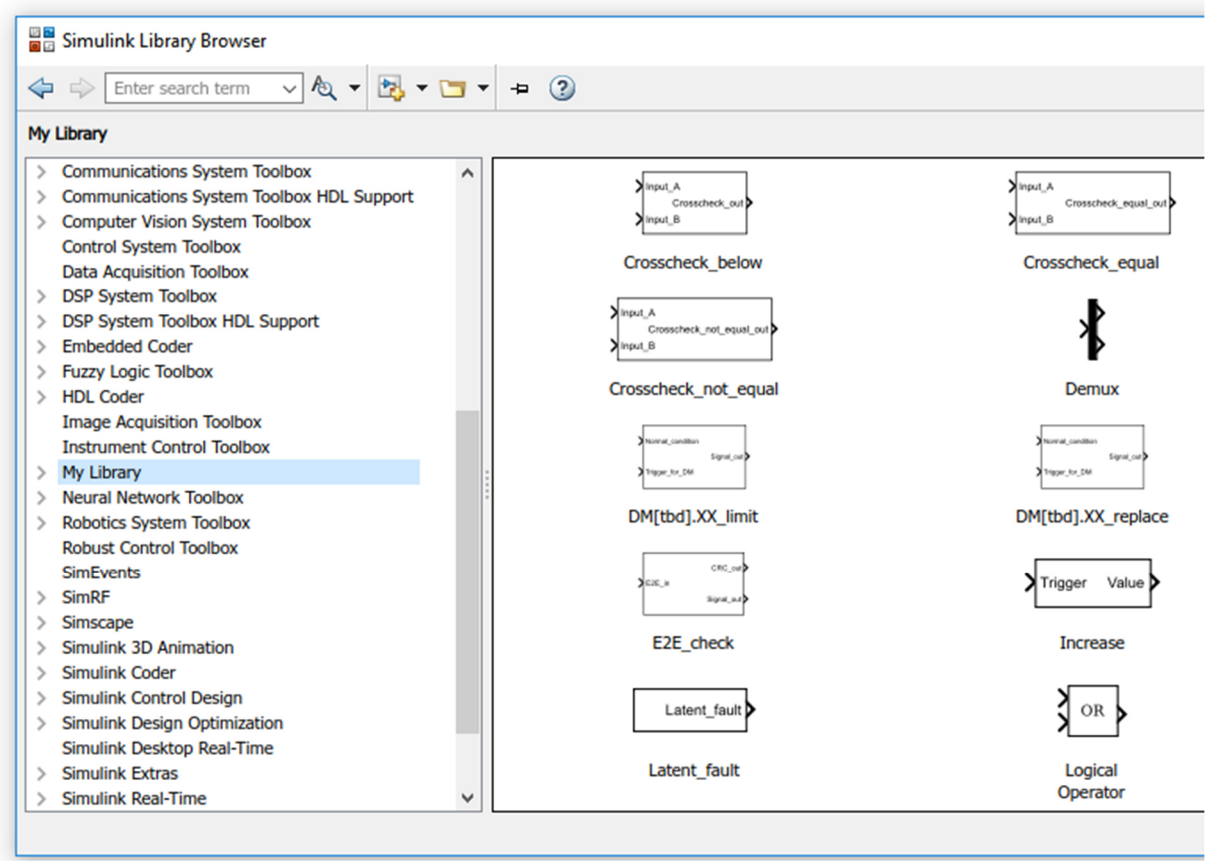

Figure 5.4: Part of the Matlab/Simulink Library

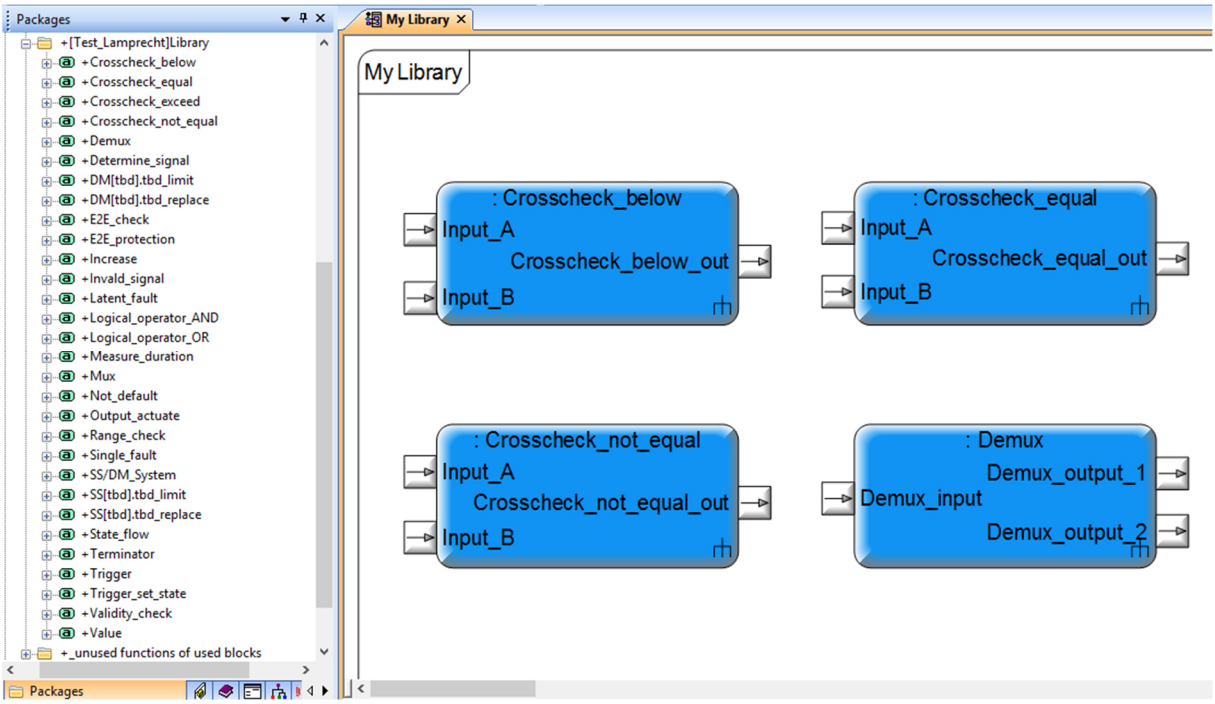

Figure 5.5: Part of the Integrity Modeler Library

In the Figures 5.5, 5.6 and 5.7 an overview of all blocks is given. The white blocks are created in Simulink and the blue blocks are the equivalent blocks in Integrity Modeler.

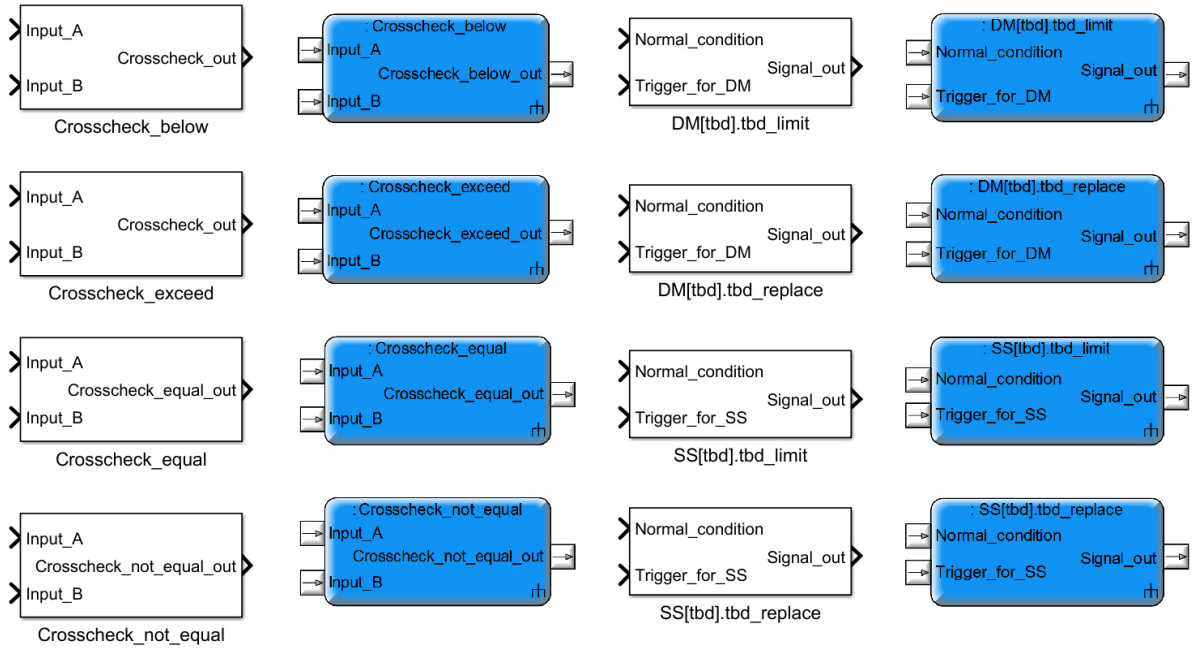

Figure 5.6: Examples of Crosscheck, Degradation mode and Safe state blocks

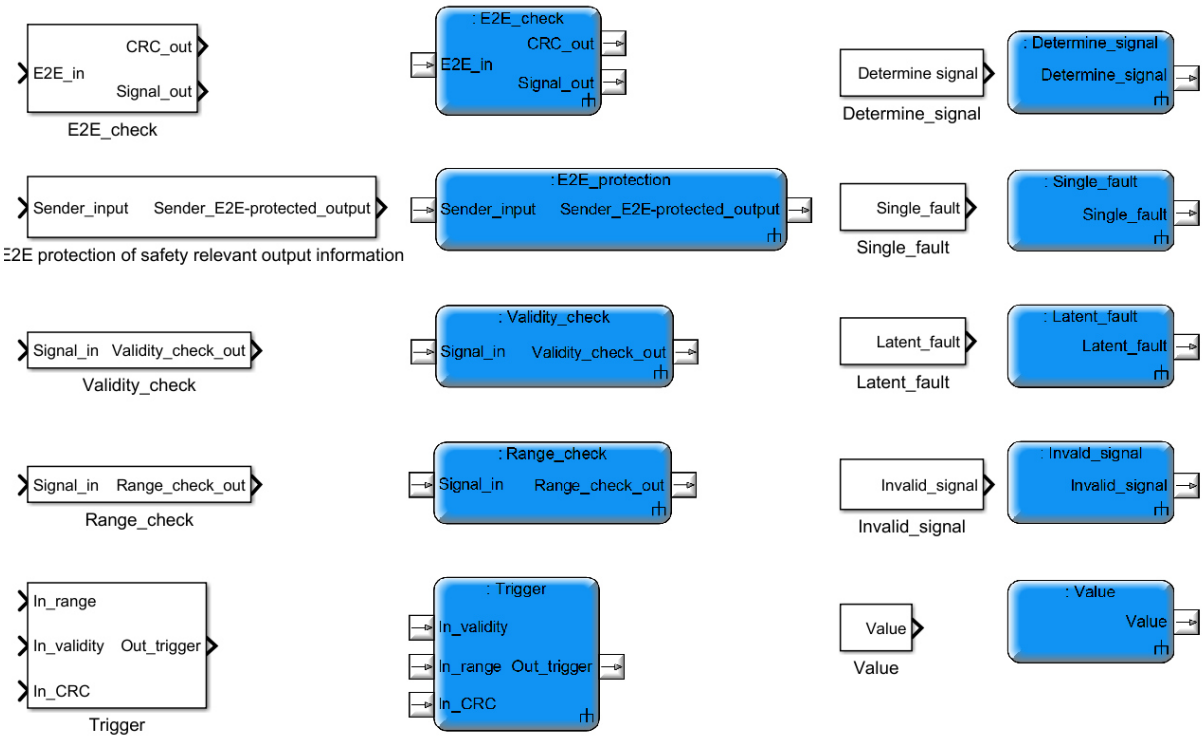

Figure 5.7: Examples of E2E blocks and Input blocks

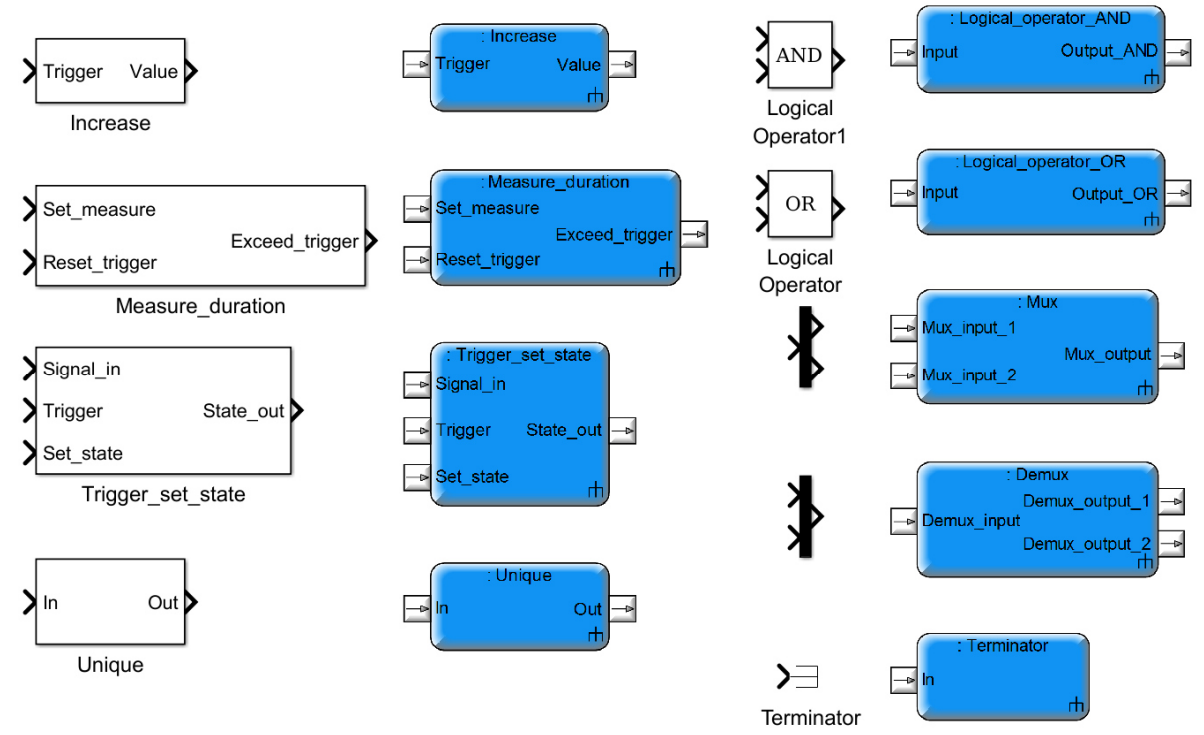

Figure 5.8: Examples of other blocks, the Unique block and standard blocks

In the following subsection the library blocks are described in detail. The blocks are separated in standard blocks and developed blocks. Where necessary and to increase the basic understanding as well as for a better comparison, the blocks in SysML will be shown too.

5.2.1 Standard Blocks

- Input/Output

These blocks and pins, respectively, are used to transfer signals between systems and subsystems. The left part of the figure represents the “Input” and “Output” blocks, as commonly used in Simulink. The right part of the figure shows the pins, which fulfill the same functionality in SysML, but are more closely related to ports than to blocks. The functionality of pins in SysML will be discussed as one of the main differences in usability of the two software packages in Section 5.4.

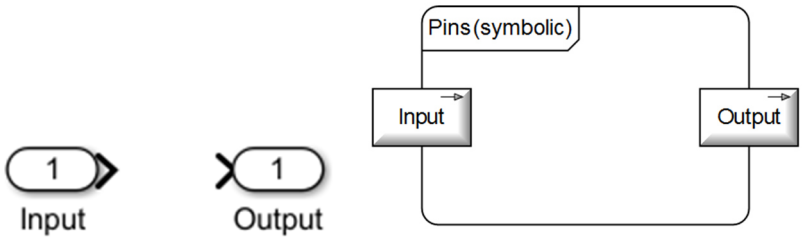

Figure 5.9: Ports in Matlab/Simulink and Pins in SysML

- Value

This block simply offers the possibility to add a constant value to the model. The value can be directly entered via an input mask, or by entering the name of a constant, which is linked to a database. The linkage to a database is a future option and was not implemented.

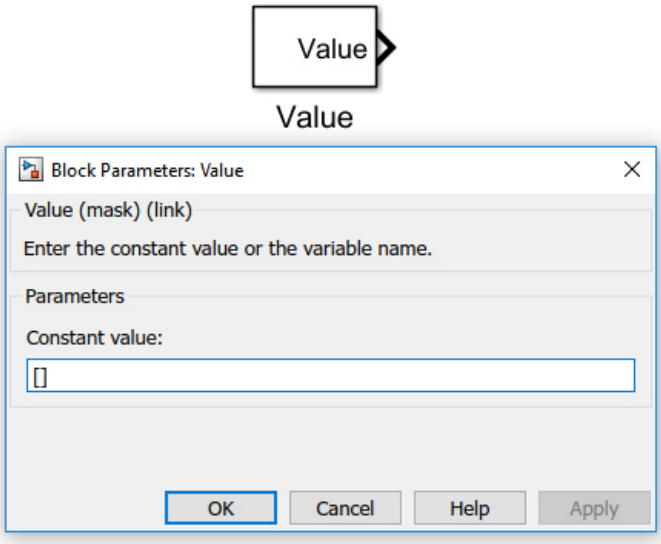

Figure 5.10: Value block

- Logical operators

Such blocks provide a variable number of input ports and trigger the output port according to logical "OR" and "AND" operators, respectively. The figure shows the blocks in Simulink (top) and in Integrity Modeler (bottom). As mentioned earlier, all blocks in Integrity Modeler are developed blocks. To improve the clarity the blocks are also presented in this section.

The blocks in both tools differ, by the ports and pins. The Simulink blocks require a port for every single input, whereas the Integrity Modeler blocks only use one pin.

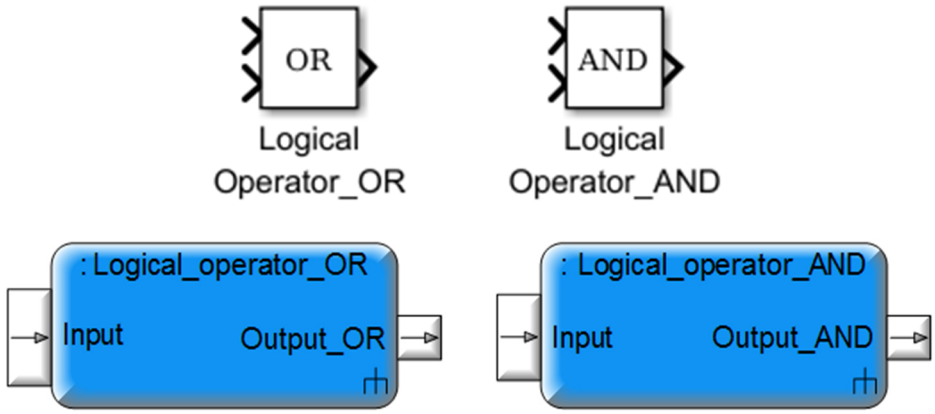

Figure 5.11: Logical operator blocks

- Mux/Demux

The function of these blocks is the combining of several signals into a virtual vector, or splitting the vector into several output values, respectively. Although this might be solved differently in an executable model, this is considered sufficient to study the feasibility of a model-based approach. The left part of the figure shows the Simulink realization with standard blocks, whereas the right part of the figure shows the SysML solution.

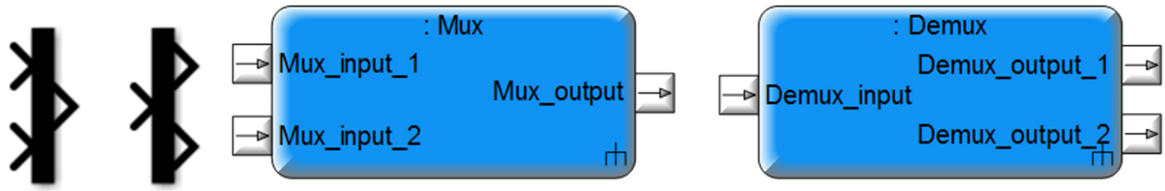

Figure 5.12: Mux and Demux block in Matlab/Simulink and Integrity Modeler

- Terminator

In case of modeling a part of a functional safety concept, some of the signals are not used in every element. Therefore, the terminator block is used. This block should symbolize that a signal exists, but it is not in use and its use was not forgotten in the model.

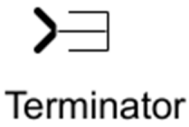

Figure 5.13: Terminator block

- Fork node

The “Fork node” is a standard block in SysML [36], which fulfills a similar function as a Simulink junction. The signal is being split up. A junction in Simulink is not a block, but it is displayed in the following figure for clarification reasons on the left part of the figure.

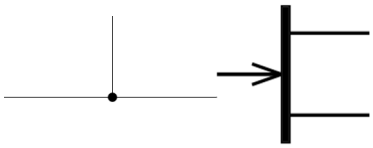

Figure 5.14: Junction and Fork node

5.2.2 Developed Blocks

The blocks mentioned in this section were developed because no known blocks in Simulink can fulfill the desired functionality. However, existing standard blocks and code in Matlab were still used to define the following blocks. Moreover, input masks were developed to offer an interface for the engineer. These interfaces offer a description of the block functionality and input boxes to define block specifics, e.g. upper or lower limits. Regarding the implementation of an input mask in Integrity Modeler, cuts were made due to the lack of the corresponding functionalities. This matter will be explained in Section 5.4.

- Determine signal

The block shown in the following figure is used to define requirements regarding the communication interface between the model and the physical existing elements. For example, this block is used to model the determination of the vehicle speed via the ESP, which is fed to the system. Additionally, an input mask offers the possibility to define the periodicity of determination. The following figure displays the block and the input mask.

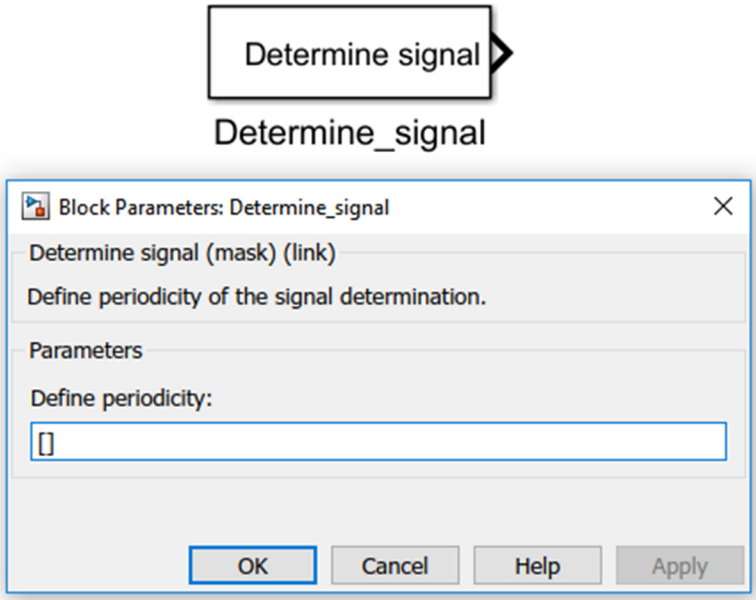

Figure 5.15: Determine signal block

- Invalid signal

This block is used to model functional safety requirements, which define the necessity to offer an interface to physical elements, so these elements can provide information about being in a problematic state. For example, information about the vehicle speed, which is provided by the ESP, is not trustworthy. The block is shown in the following figure. The input mask to define the report time is similar to the one, shown in Figure 5.15.

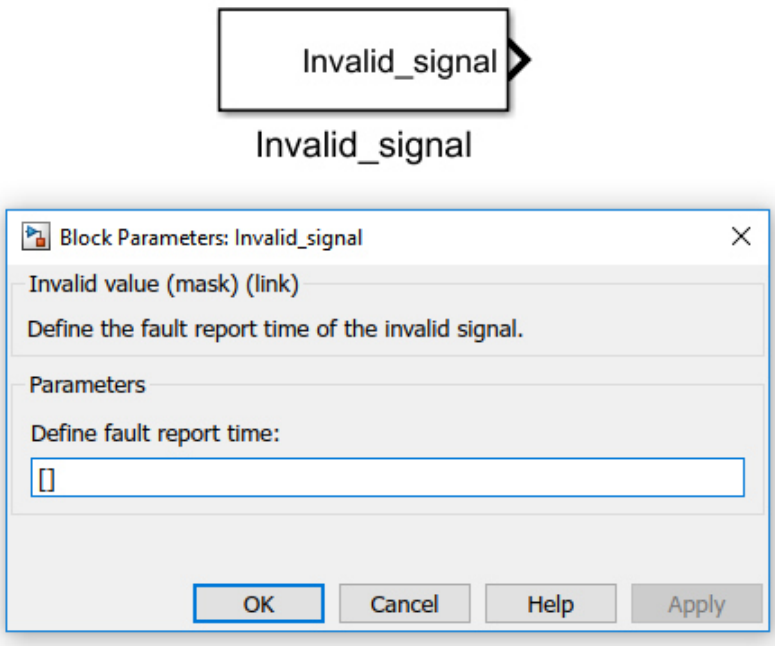

Figure 5.16: Invalid signal block

- Single/Latent fault

Like the “Determine_signal” block, these two blocks represent an interface for physical elements to inform about failures, which are detected internally and cause safety violations in absence of safety mechanisms. Furthermore, the blocks offer an input box for periodicity, communication time and the option to enter specific textual information, which is added to the requirement.

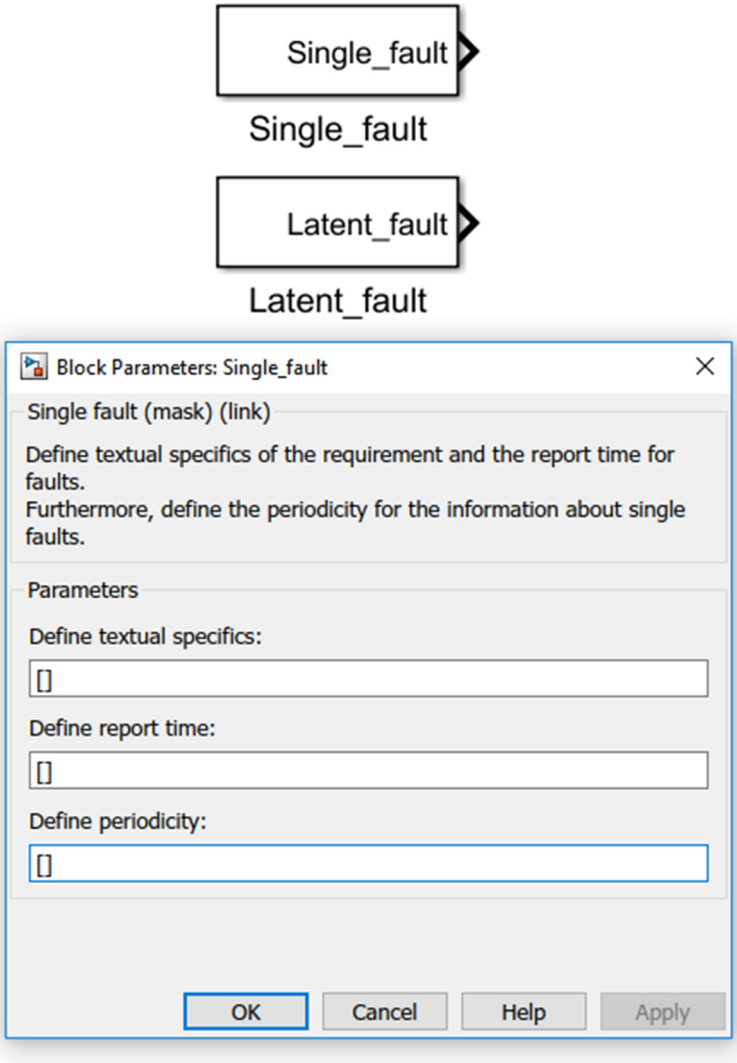

Figure 5.17: Single and latent fault blocks

- Validity check

The aim of this block is to check the input for a specific “Invalid value”, which can be defined in the input mask and if necessary, set a trigger signal. The block and the corresponding input mask can be seen in the following figure.

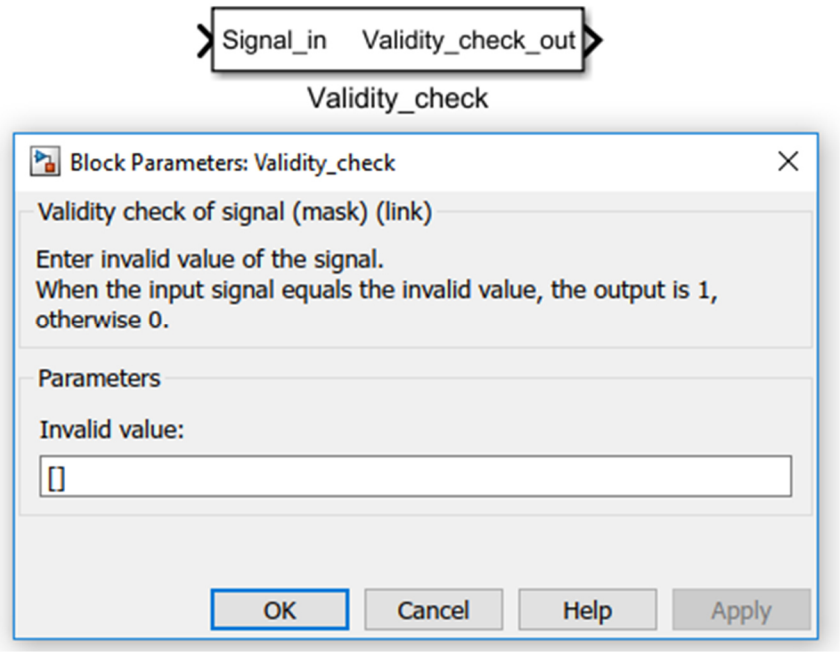

Figure 5.18: Validity check block

- Range check

This block fulfills a similar function as the previous block, but instead of a specific value, the input is checked to be in a specific range. The definitions of upper and lower limits for the range are again implemented with an input mask.

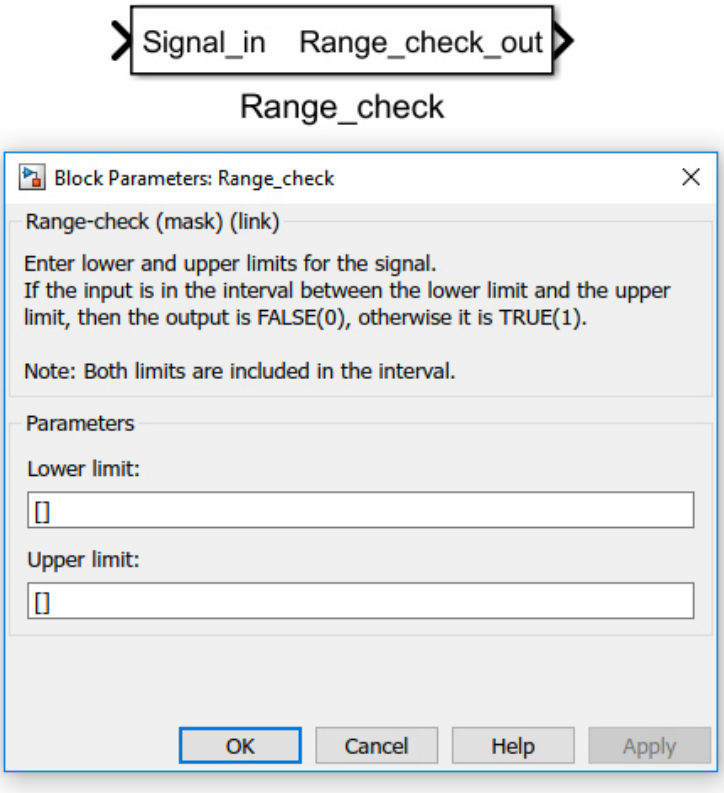

Figure 5.19: Range check block

- E2E protection of safety relevant output information

Together with the next described block, these two blocks reflect the requirements to secure the CAN-Bus based communication between different elements. The aim is to prevent the processing of wrong signals, or at least detect that the signal is not trustworthy. Hence, this block applies different mechanisms to secure the input information before transferring it via the output port to the CAN-Bus. Such mechanisms are e.g. a CRC. As these mechanisms are not part of this thesis, only a simplified function was implemented in the block. The block and the subsystem with the simplified function of a CRC can be seen in the following figures.

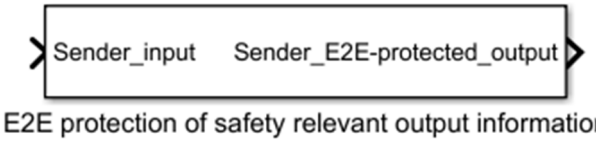

Figure 5.20: E2E protection block

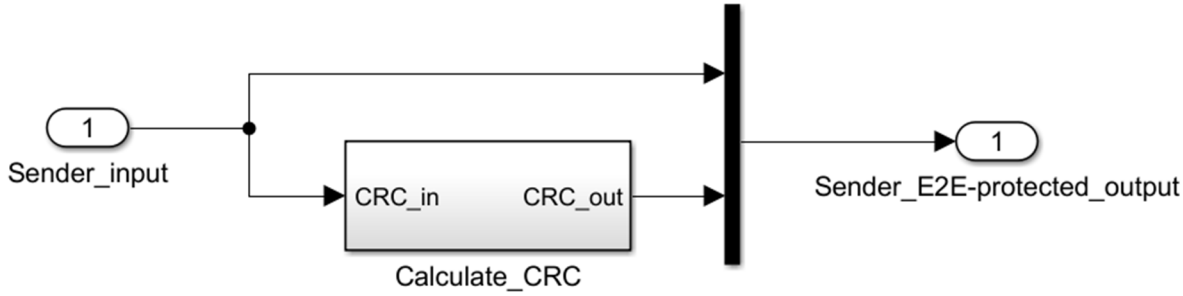

Figure 5.21: Subsystem of E2E protection block

- Monitoring of external input information (E2E check)

This block receives the E2E protected information and checks e.g. the CRC. The output is the signal itself and a trigger signal, in case the CRC is not correct. The following figures show the block and the according subsystem with the CRC verification. At this point it is important to mention that the signal itself is neither overwritten nor changed in any way. In case the communication is faulty, this fault will be handled with in-line functions and blocks, respectively.

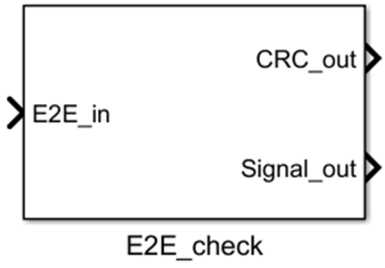

Figure 5.22: E2E check block

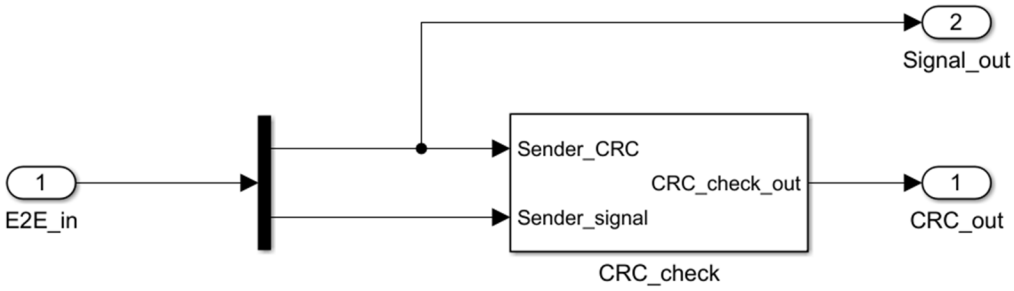

Figure 5.23: Subsystem of E2E check block

- Trigger

The following figure shows the “Trigger” block. Its purpose is the determination of a trigger signal being received from the “Range”, “Validity” or “CRC check” and to set the output trigger within a specific time, which can be defined via an input mask. The functionality is similar to an “OR” block, but differs by offering an input mask and the possibility to define the duration, between a trigger signal being received and the block triggering the output.

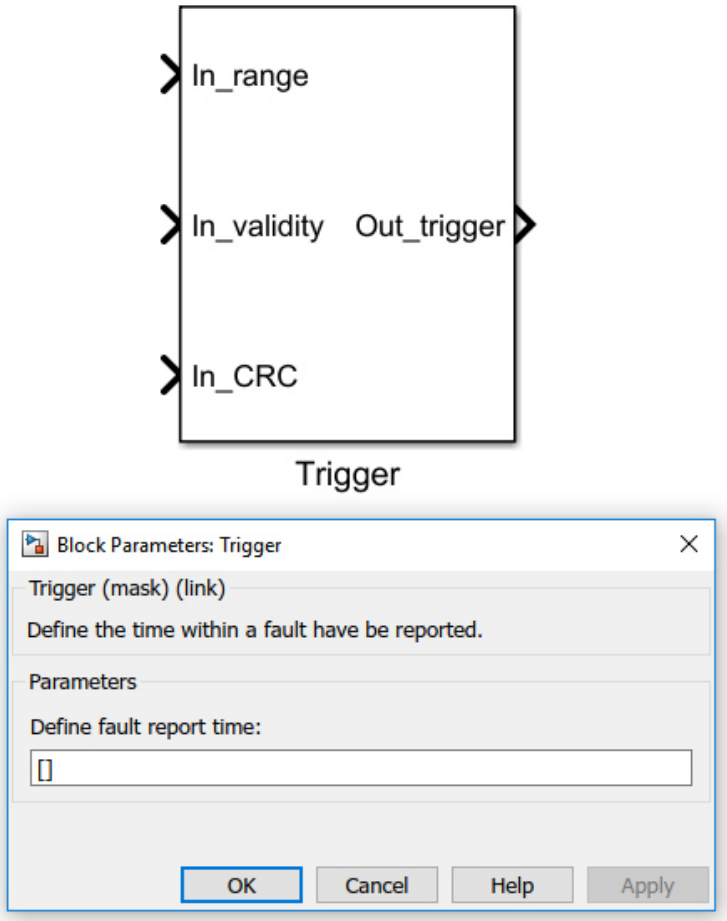

Figure 5.24: Trigger block

In Section 5.2.4, the communication between elements will be shown.

- Crosscheck (not) equal

These two blocks fulfill the requirement of comparing the input signals for (in)equality and if necessary trigger an output signal. For “Input_B” an input mask offers the possibility to define a range of tolerance. As the figure shows, an additional input box allows for the definition of the acceptable time between the condition being fulfilled for the first time and the fault being set.

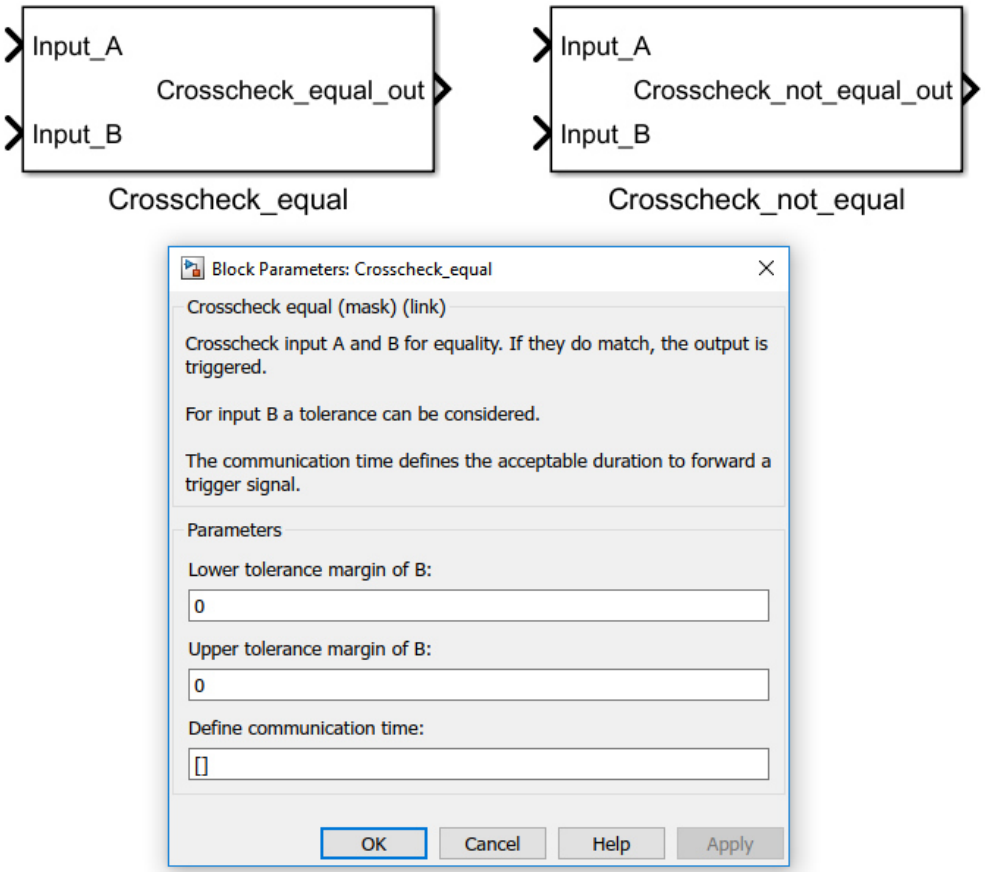

Figure 5.25: Crosscheck (un)equal block

- Crosscheck below/exceed

The two blocks shown in the following figure are owed to functional safety requirements, in which it is necessary to compare the size of two signals. In case of “Input_A” being below/exceeding “Input_B”, a trigger signal is set. Furthermore, the input mask allows the definition for the duration of the proceeding.

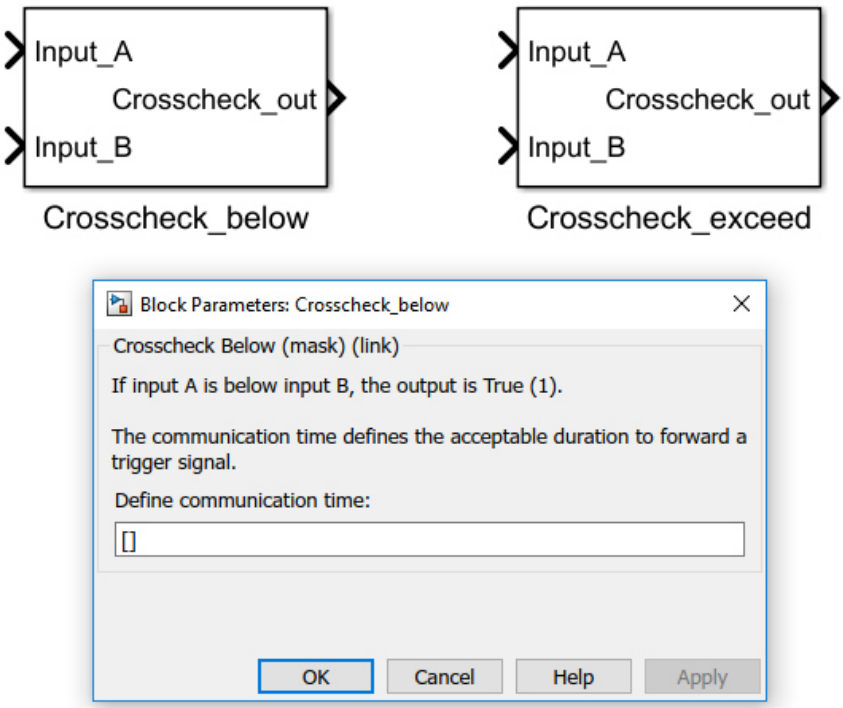

Figure 5.26: Crosscheck below/exceed block

- Trigger set state

This block fulfills the requirement of setting a specific state. In case the trigger is set, the information from the "Signal_in" port is overwritten by the information of the "Set_state" input port. The following figure shows the according block in Simulink.

Figure 5.27: Trigger set state

- Measure duration

The "Measure duration" block is owed the necessity of some functional safety requirements to define a duration, until a trigger signal is set. The block offers two input ports and an input mask. If a block receives a trigger signal, a timekeeping function is activated, which lasts either until a reset trigger signal is received via the second input port, or the duration exceeds the value, which was predefined via the input mask. If the defined duration is exceeded, the output port sends an according trigger signal.

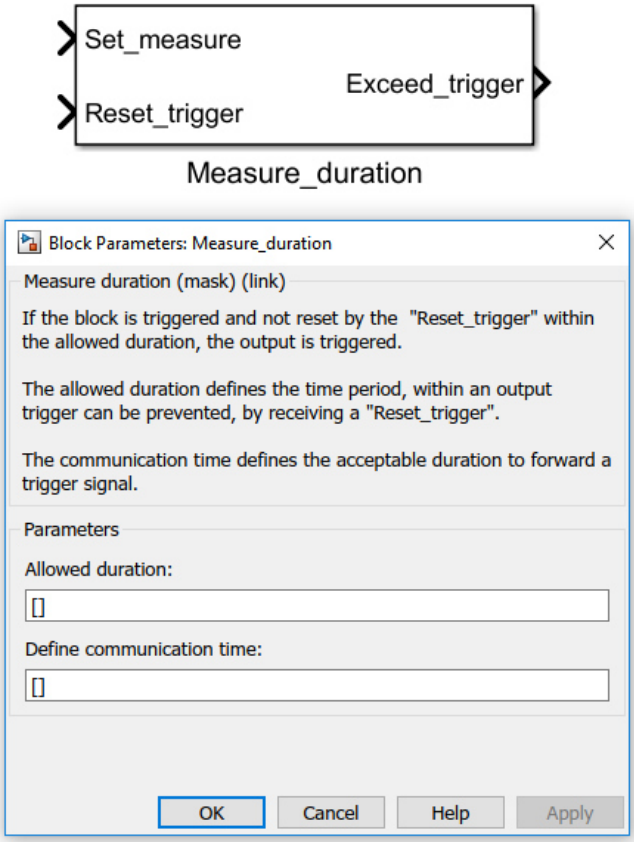

Figure 5.28: Measure duration

- Increase

The block shown in the following figure fulfills several tasks demanded by functional safety requirements. If a trigger is received, the implemented counter increases the previously saved value by one. This new value is again saved in a memory and provided to the in-line blocks via the output port. The memory itself allows storing the value beyond a vehicle restart.

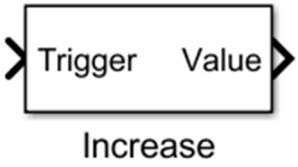

Figure 5.29: Increase block

- Safe states and degradation modes

As described in an earlier chapter, SSs and DMs are closely related and set measurements to prevent safety goal violations. Hence, the blocks are also related. If a trigger signal is received, the block's original input signal is either limited (by a DM) or replaced (by a SS). The input possibility for limitation and replacement values is again implemented with an input mask. An additional input box in these masks allows the definition of the allowed duration between the condition being fulfilled and the fault

being set. The distinguishing feature of these blocks compared to others is the implementation of a latch function. As a result, the output signal is only reset when the system is being restarted.

Figure 5.30 shows the four blocks and Figure 5.31 shows the input mask for a SS limitation block.

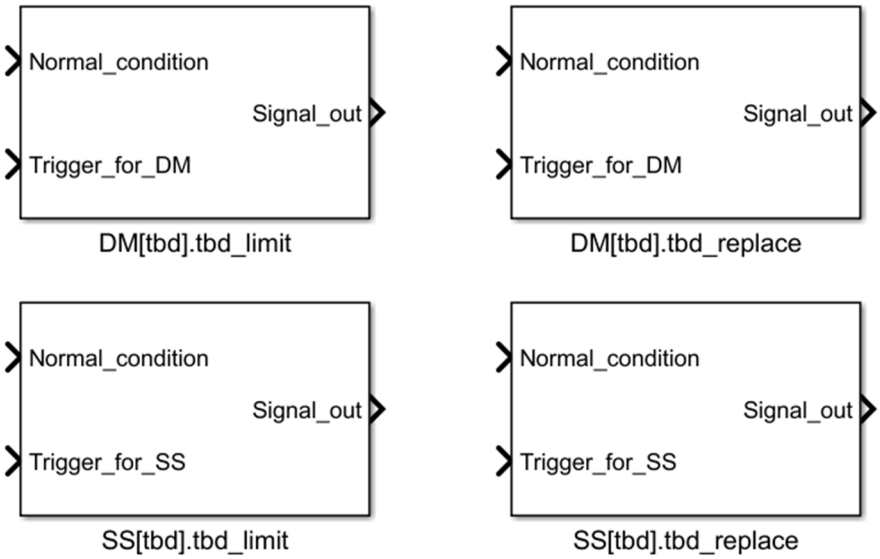

Figure 5.30: Safe state and degradation mode blocks

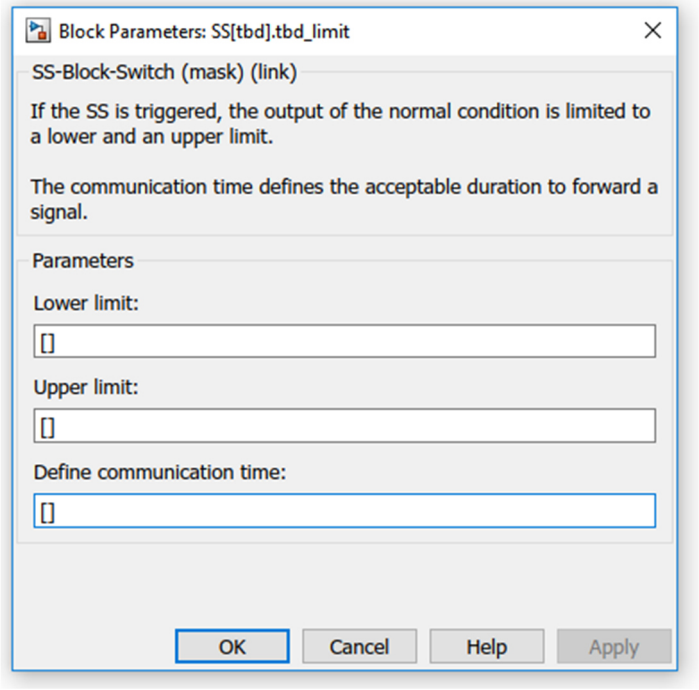

Figure 5.31: Input mask for SS blocks

5.2.3 Block for Unique Purposes

Some functional safety requirements cannot be modeled with the previously presented library blocks, but to prevent an incomplete model, these requirements needed to be modeled too. Hence, the following block is used.

- Unique

This block is used to model functional safety requirements, or parts of functional safety requirements, which cannot be expressed by any of the previous mentioned blocks or block combinations and shall function as a wild card to design an individual solution in the model. An example for the use of this block can be found in Section 5.2.5.

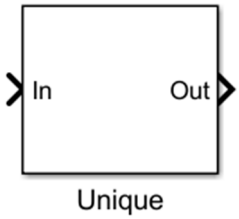

Figure 5.32: Unique block

5.2.4 Examples of Requirements Modeled with Blocks

The following section will give an insight in the transfer of the text-based functional safety requirements into block models. The basis for each example is an ESM of the functional safety concept.

- ESM[HVBATT].212 Vehicle crash report handling

In case of a vehicle crash, this ESM of the HV battery should prevent the battery contactors from being closed, or open them if they are already closed by triggering relevant safe states.

The following two figures show the according functional safety requirements and the equivalent Simulink model with blocks, which shall reflect the textual information of the requirement. For a better visualization is the “REQ-ID” of every functional safety requirement an accordingly named area in the model. These areas are usually not created in the model.

[**REQ-ID**: 1184251] [**State**: External Review] [**ASIL**:ASIL A] [**Category**: Functional Requirement] [**Allocation**: 1060141]

[06] IF
§1 the vehicle crash information indicates a crash 1§
THEN
the §2 HV Battery 2§ shall §3 remember the occurrence of this event by storing it persistently (e.g. in the NVRAM) 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1184251]

[**REQ-ID**: 1427334] [**State**: In Work] [**ASIL**:ASIL A] [**Category**: Functional Requirement] [**Allocation**: 1060141]

[02] IF
§1 ((the vehicle crash information received via the vehicle communication bus indicates a crash) AND
(HCU has not requested HV disconnection within {{SafC_TiEdriveReactTol}}) 1§
THEN
the §2 HVBATT 2§ shall trigger §3 SS[HVBATT].01 and SS[HVBATT].02 3§ within §4 {{SafC_TiFltRctn2ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1427334]

[**REQ-ID**: 1185856] [**State**: External Review] [**ASIL**:ASIL A] [**Category**: Functional Requirement] [**Allocation**: 1060141]

[02] IF
§1 a crash is indicated by the stored vehicle crash information at startup 1§
THEN
the §2 HV Battery 2§ shall trigger §3 SS[HVBATT].05 and SS[HVBATT].02 3§ within §4 {{SafC_TiFltRctn2ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1185856]

Figure 5.33: Vehicle crash report [4]

The first functional safety requirement of Figure 5.33 can be fulfilled with a block, comparing two values for equality. On the one hand a predefined value and on the other hand the received crash information. If they match, a trigger signal is stored.

The second requirement of Figure 5.33 necessitates information about the equality of two signals. In case of a certain duration of the information being unequal, a trigger is set. Furthermore, a trigger from an occurred crash is necessary.

The third requirement also demands comparison of two signals and sets an according trigger.

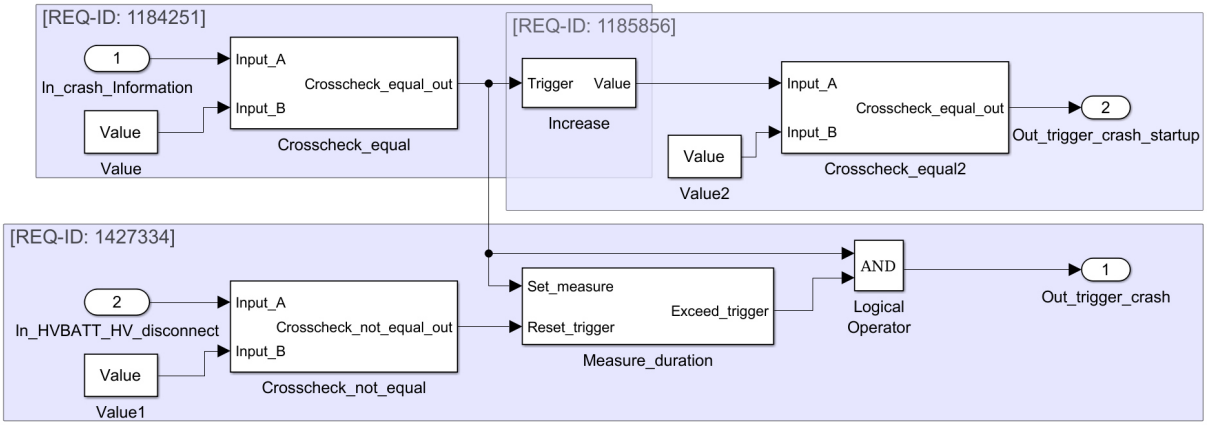

Figure 5.34: Vehicle crash report model

- ESM[ESP].300 E2E protection of safety relevant output information

  ESM[HCU].100 Monitoring of external input information (E2E check)

These ESMs from the ESP and the HCU are typical for the information transfer between two elements. In the following two figures an example of the information transfer for the vehicle speed is shown. The vehicle speed is part of the CAN-Bus information provided by the ESP and this information is received e.g. from the HCU. Therefore, the vehicle speed is E2E protected in the ESP and afterwards the signal is checked in the HCU for its correct transmission. Like in the previous example, the "REQ-ID" of the functional safety requirements is the same as the emphasized areas in the block model.

ESP:

[**REQ-ID**: 1152353] [**State**: External Review] [**ASIL**:ASIL C] [**Category**: Functional Requirement] [**Allocation**: 1161533]

[01] The §1 ESP 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForEsp}} 4§ at maximum.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1152353]

HCU:

[**REQ-ID**: 1147337] [**State**: External Review] [**ASIL**:ASIL C] [**Category**: Functional Requirement] [**Allocation**: 1060149]

[01] The §1 HCU 1§ shall §2 perform an end-to-end check on all safety relevant input communication signals 2§ periodically each §4 {{SafC_TiDetn1CanInpInfo}} 4§ at maximum.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1147337]

[**REQ-ID**: 1168299] [**State**: External Review] [**ASIL**:ASIL C] [**Category**: Functional Requirement] [**Allocation**: 1060149]

[02] IF
§1 an E2E failure of the Vehicle speed information is detected 1§
THEN
the §2 HCU 2§ shall trigger §3 SS[HCU].01 3§ within §4 {{SafC_TiFltDetn1}} 4§ and shall set §3 HCU_Vehicle_speed_external_fault 3§ within §4 {{SafC_TiFltDetnForSetFlt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1168299]

Figure 5.35: Example of E2E communication [4]

The first functional safety requirement of Figure 5.35 is part of the ESP and demands that the sender signal is E2E protected. This is executed by the according block, which is described in Figure 5.36. As mentioned earlier, the E2E protection is modeled in a very simplified way.

The second functional safety requirement of Figure 5.35 is part of the HCU and demands the received signal to be checked for the correct range, validity and CRC. And the third functional safety requirement requires a trigger signal, which sets an E2E fault and an additional fault state. These two requirements are modeled in Figure 5.37. The additionally set fault state is not further used in this model and therefore terminated.

An important side note is the fact that although the vehicle speed signal might be faulty, the receiver still forwards the faulty signal, but sets additional trigger signals for e.g. safe states, which cause safety mechanisms in the subsequent elements.

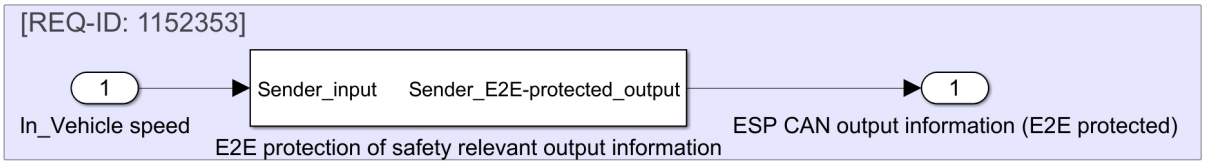

Figure 5.36: Example of E2E communication model of the ESP

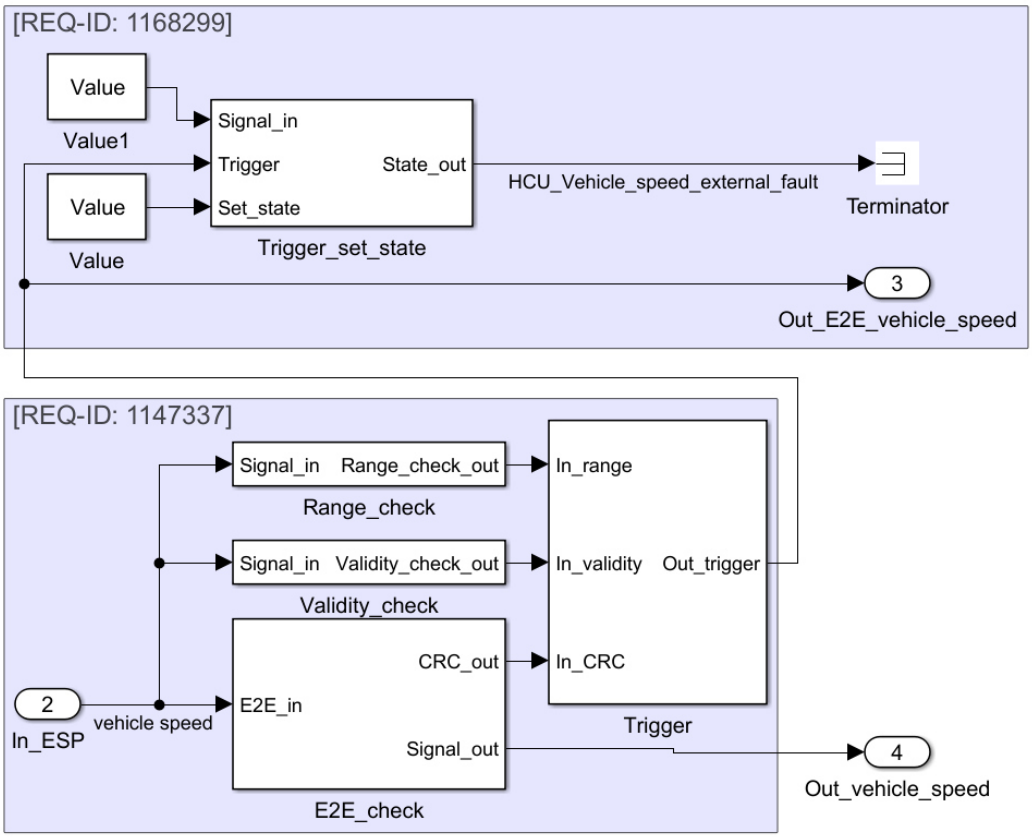

Figure 5.37: Example of E2E communication model of the HCU

5.2.5 Example for Unique Blocks

- ESM[HVBATT].208 HV contactor weld check

The purpose of this ESM is the detection of welded contactors.

The according functional safety requirements in Figure 5.8 are a typical example for requirements, which cannot be modeled with standard blocks. This requirement does not fit any block. Therefore, the unique block is used in the model. The second requirement indicates the use of a simple crosscheck block. Like in the previous examples, the “REQ-ID” of the functional safety requirements match the area of the model in Figure 5.39.

[REQ-ID: 1185777] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060141]

[01] The §1 HV Battery 1§ shall §2 check the HV battery contactors and the precharge contactors considering the
(a) the DC link voltage and
(b) the HV battery pack voltage
2§ in case of §3 opening or closing the HV battery contactors 3§ within §4 {{SafC_TiFltDetnForHvbatCtctr}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1185777]

[REQ-ID: 1185779] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060141]

[02] IF
§1 weld check indicates a critical fault/failure 1§
THEN
the §2 HV Battery 2§ shall trigger §3 SS[HVBATT].02 and SS[HVBATT].05 3§ within §4 {{SafC_TiFltRctn2ForSSt}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1185779]

Figure 5.38: Example of unique requirement [4]

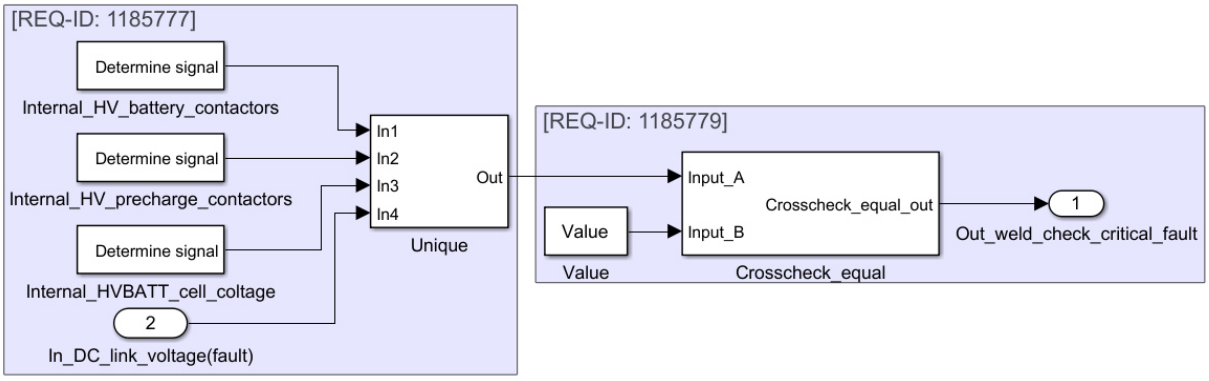

Figure 5.39: Example of unique blocks

5.2.6 Statistical Results

In this section some statistical results regarding the model of SG07 are presented to accentuate the found patterns and potential for simplifications with predefined blocks.

72 of the 183 ESMs examined were identified to be relevant for SG07. The relevant element safety mechanisms are described with 240 functional safety requirements, from which again only 160 were relevant to describe SG07. The rest of the element safety mechanisms and functional safety requirements were necessary for the description of other safety goals.

The mentioned 160 requirements were described with about 400 blocks. The interesting finding is that only approximately 5 % of the functional safety requirements, required a block for unique purposes. This indicates that a lot of the functional safety requirements can be modeled with predefined blocks. This finding is presented in the pie chart of Figure 5.40. To improve the clarity in Figure 5.40, the blocks

were grouped according to their purpose, e.g. blocks like “Crosscheck exceed” and “Crosscheck equal” etc. are put together to one group named “Crosscheck”.

Another interesting finding is that the E2E protection of signals with blocks like “E2E protection of safety relevant output information” and “Monitoring of external input information (E2E check)” sum up to almost ⅓ of all used blocks. Another remark refers to the blocks for crosschecks. More than ⅕ of all blocks are part of this group, of which the purpose is the comparison of two signals.

An interpretation of these results is shown in Chapter 7.

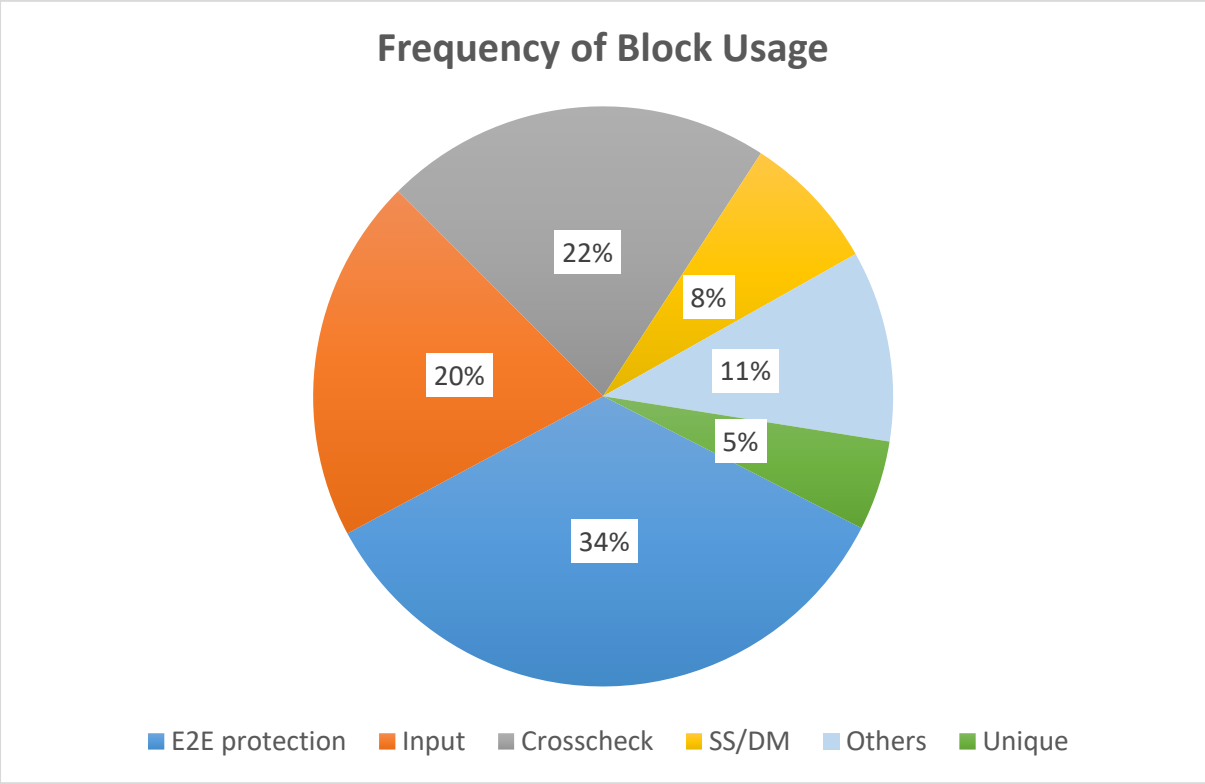

Figure 5.40: Statistical result of used blocks

A clear causal relation between the used requirement sentence template type and the used blocks were not found. The reasons are the templates and the developed solution. Some templates are used for similar purposes and to derive a block-based solution, the functional safety requirements created with the templates are split in parts. As a result, are the requirements which are derived from various templates, modeled with many block combinations and this does not allow the definition of clear causalities.

In case of the example, shown in Figure 5.41, are two different templates used, but the first part of both requirements is modeled with the same block type (“Crosscheck below”), because the signal word is “below” (red).

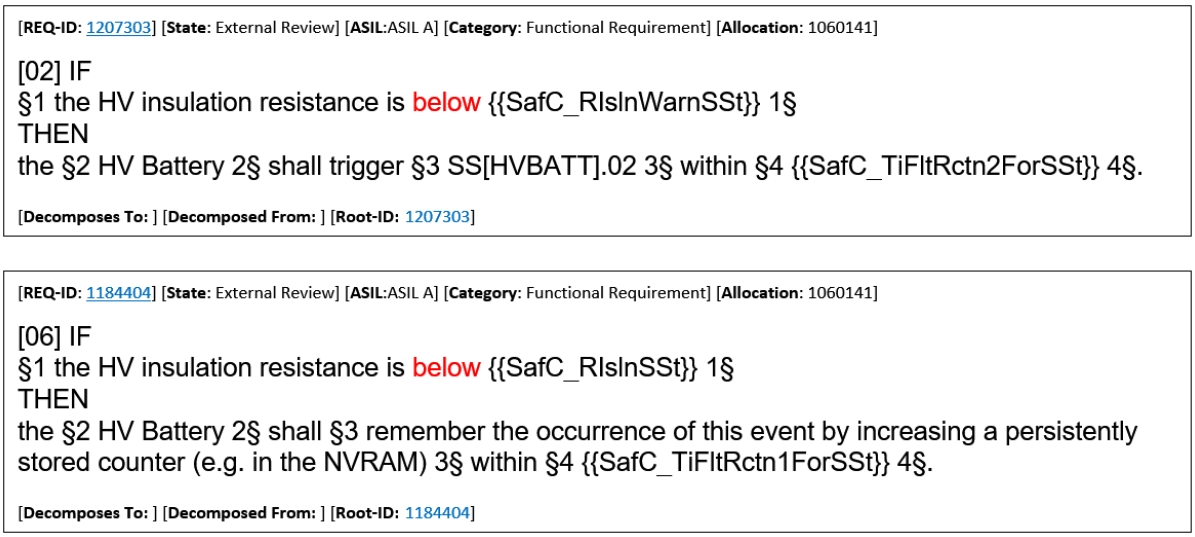

Figure 5.41: Example for requirements with different templates and same blocks, derived from [4]

5.3 Simplifications

Due to the conceptual approach, several simplifications of the model were necessary. These simplifications will be explained in this section and arguments for it will be mentioned. Some of the described decisions are based on boundaries of this thesis, which were described in Section 1.

- The first simplification is based on the fact that only a part of the whole functional safety concept was modeled. Therefore, blocks like the “Terminator”-block, the “Determine signal”-block and the “Input”-, “Output”-blocks and the pins, respectively, were used to limit the model boundaries and ensure completeness.
- To increase the comprehensibility for future developments, the ports and signals were named to match their application. This is problematic because the software development domain demands the use of defined nomenclature, but it was decided by the author of this thesis to be accepted at this conceptual level, because the comprehensibility was considered of higher priority.
- All blocks provide basic functionalities like input masks for specifications. Executable functionality which would require more extensive coding, or the implementation of e.g. specific data bases were not in the scope of the thesis.
- The E2E protection and check of CAN-Bus signals are simplified in the model and do not fulfill any defined or required specifications of a real model. This was done because the security of the communication was not a topic of this thesis.

  A E2E check should cover “message corruption”, “message delay”, “message loss” etc.

  In case of this thesis was the E2E protection only indicated by a CRC

- The signal routing with blocks like “Mux” and “Demux”, and the “Fork node” in Integrity Modeler is also simplified, because it was also not in the focus of this thesis.
- The HVIL is a hard-wired electrical component of the vehicle, which was modeled, due to simplicity reasons, to be determined at several points of the model. In the actual vehicle it is a closed loop, the interruption of which leads to a failure reaction. The HVIL and this simplification are shown in Section 6.
- Figure 5.42 shows the general content of each functional safety requirement (red). This part of the requirements is not taken into consideration for the developed blocks, because it was considered of secondary relevance for a first model-based approach. One possible implementation can be derived by input masks, similar to the information about periodicity and duration.

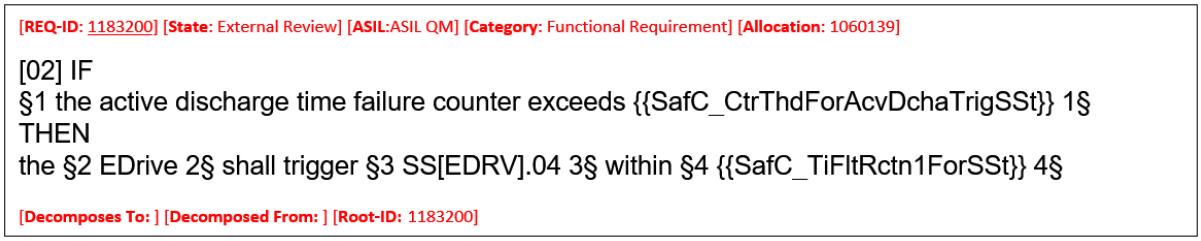

Figure 5.42: Example for general content of requirements, derived from [4]

5.4 Software Comparison

In this section some of the differences regarding the use of Simulink and SysML in Integrity Modeler, which were found during use, are described. Due to the difference of SysML in Integrity Modeler, being a general modeling language and Simulink being a graphical programming and modeling environment, a comparison of the extensive functionalities of both tools would exceed the scope of this thesis. The aim of this section is not to give any advice on which software tool should be chosen or which tool should be preferred, but it should provide some insights in the challenges engineers will face using the library in one or the other software tool. Still it is important to mention that although the tools differ in many ways, both fulfill the key aim of this thesis, which is the creation of a reusable library with predefined blocks.

Differences between the tools and modeling languages:

- The first key difference of both tools is the model itself. Simulink offers the possibility to create an executable model with interfaces to Matlab and data bases. SysML in Integrity Modeler does not offer this possibility. But it is a modeling language with a predefined semantic and different diagram types for different purposes, and it allows the modeling of the structure of a system separately from its behavior [36]. The Simulink’s structure and behavior are in the same model,

which can get confusing in bigger models. Nevertheless, allows Simulink the creation of different levels.

- Once the blocks in the library are created, the Simulink library offers a list of blocks, a block preview and a help function. Integrity Modeler only offers a list of the blocks, but in return are all the applied sub blocks and pins easily visible in the directory tree. As a result, is the Simulink library more comprehensible, but the access speed to the Integrity Modeler library is higher.
- Simulink provides the possibility to mask every block and prevent the modification of the block's properties. On the other hand, Integrity Modeler offers the possibility for changes on every block in the library and more adaptations in the model.
- With Simulink it is possible to create a predefined input mask for every block, which can offer descriptions and be filled with block specifics during modeling. The inputs can be visualized in the model and connected in the subsystem of the block to e.g. a database for further specification. In contrary to this, Integrity Modeler offers the possibility for extensive notes and descriptions in the properties of every individual block, but no input masks.
- Every block in Simulink which is used in the model inherits its size, number and position of ports from the library. The number of ports can also be adjustable. Integrity Modeler offers no predefined design and this makes it necessary to format every single block, whenever the block is used. On the other hand, the position and number of pins, which is the equivalent of the ports in Simulink, is fully variable and every pin can be sender or receiver, respectively, for several signals.[14]

  The following Figure 5.43 shows an example of the "E2E_check", after it was positioned in the model. As described, the Simulink block is predefined, whereas the SysML block must be formatted, before the final design, like in Figure 5.44, is accomplished.

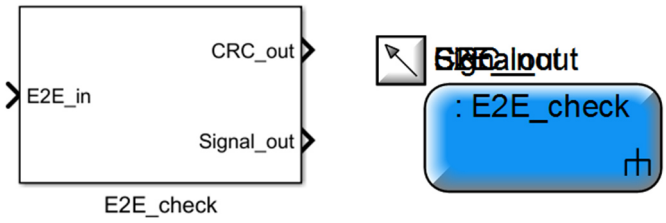

Figure 5.43: Block formatting in both tools

[14] Due to comparability of both models, these possibilities of SysML were only used for the logical operator blocks.

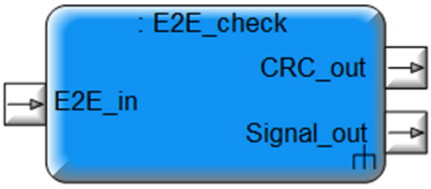

Figure 5.44: Formatted “E2E_check” in Integrity Modeler

- Between systems and subsystems and vice versa, Integrity Modeler allows passing on pins and makes clear which pins have not been used yet. Furthermore, an automatic populate function exists. Contrary to this, Simulink passes on all ports automatically from the subsystem to the system, but not vice versa. Every port must be numbered and the port number in the subsystem defines the position of the port in the system above.
- For a comprehensible model and as a modeling guideline at AVL, straight signal lines are necessary. Simulink offers automatically straight signal lines. Moreover, automatic connection of block ports and an automatic routing of the signal lines, even if the position of blocks has been changed, is implemented in the functionality of Simulink. In Integrity Modeler object flows are the equivalent. For object flows, straight lines are not the standard and at every bending of the signal line, this must be considered. Furthermore, if a block position is being changed, every signal line must be adapted. On the other hand, Integrity Modeler does allow a very exact positioning of every object flow.
- To split a signal in Simulink, a single click is enough. In contrary to this, SysML as modeling language allows two ways to fulfill this task. Either two signals leave the output pin, or a “Fork node” is used. This difference is also shown in the corresponding block in Section 5.2.1. The disadvantage of the “Fork node” is its orientation. To change the orientation of the “Fork Node”, the view options must be opened in a separate window, which is far more work than a junction in Simulink.
- Another difference between Simulink and Integrity Modeler is the support for copy and paste operations. Simulink allows these functions without the necessity for adaptation of the blocks afterwards. In contrary, Integrity Modeler does allow copy and paste operations, but every copy of a block must be formatted again.
- Another advantage of the Integrity Modeler library is the creation and adaptation of blocks. Blocks and pins can be added and removed very fast within the directory tree. The access to implement modifications in the Simulink library is more complicated.

6 HV-SG07: AVOID UNINTENDED HV

In this section the modeled SG07 will be explained in more detail. This gives one an extensive understanding of the block library, the created model and the functional safety concept it is based on. Most of the figures will only show the Simulink model. The model in Integrity Modeler is only presented where necessary, or to get a better understanding of the model. Other safety goals of the functional safety concept, like the mentioned SG08 and SG09 will not be taken into consideration, because this would exceed the scale of a simple example. All the requirements are parts of the functional safety concept and therefore based on information which is provided by AVL.

As aforementioned is the purpose of SG07 the prevention of high-voltage incidents, which could harm humans. According to the functional safety concept is this achieved by a voltage reduction or even separation of the high voltage parts (HVBATT, HVAC, OBC, EDRV, DCDC. An overview and description of the architecture of the P2-HEV and all involved elements can be found in Figure 6.1. The elements which are involved in SG07 are within green rectangles.

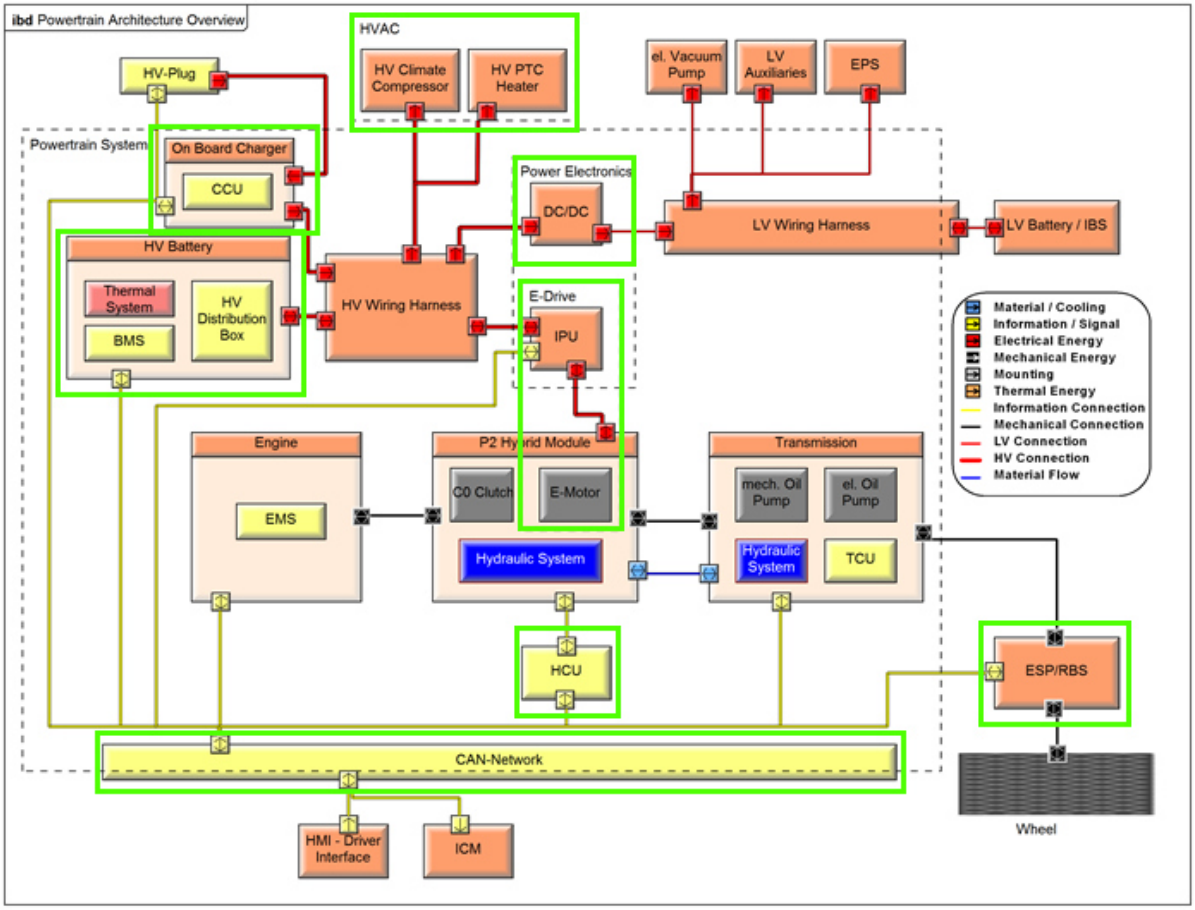

Figure 6.1: Elements involved in SG07, derived from [4]

The elements which are part of the SG07 fulfill very different tasks. The purpose of the OBC is the provision of electrical energy from the HV-plug to the HVBATT. The HVBATT stores electrical energy and provides it to the vehicle systems. Especially the supply of the EDRV (E-Drive) is an important task of the HVBATT. The EDRV converts the electrical energy into mechanical energy. The purpose of the HVAC is heating, ventilation and air conditioning. The DCDC is an electric converter from high voltage to low voltage. The ESP improves the stability of the vehicle and in case of SG07, provides information like the vehicle speed. The HCU controls all the systems via the CAN-Network.

To prevent the violation of SG07, voltage reduction or separation of high voltage elements is triggered on powertrain level by either an HVIL interruption, a crash, or a problematic insulation resistance. The PTSMs of SG07 are defined with four requirements, reflecting these trigger reasons.

The requirements of SG07 on powertrain level are shown in the following figure.

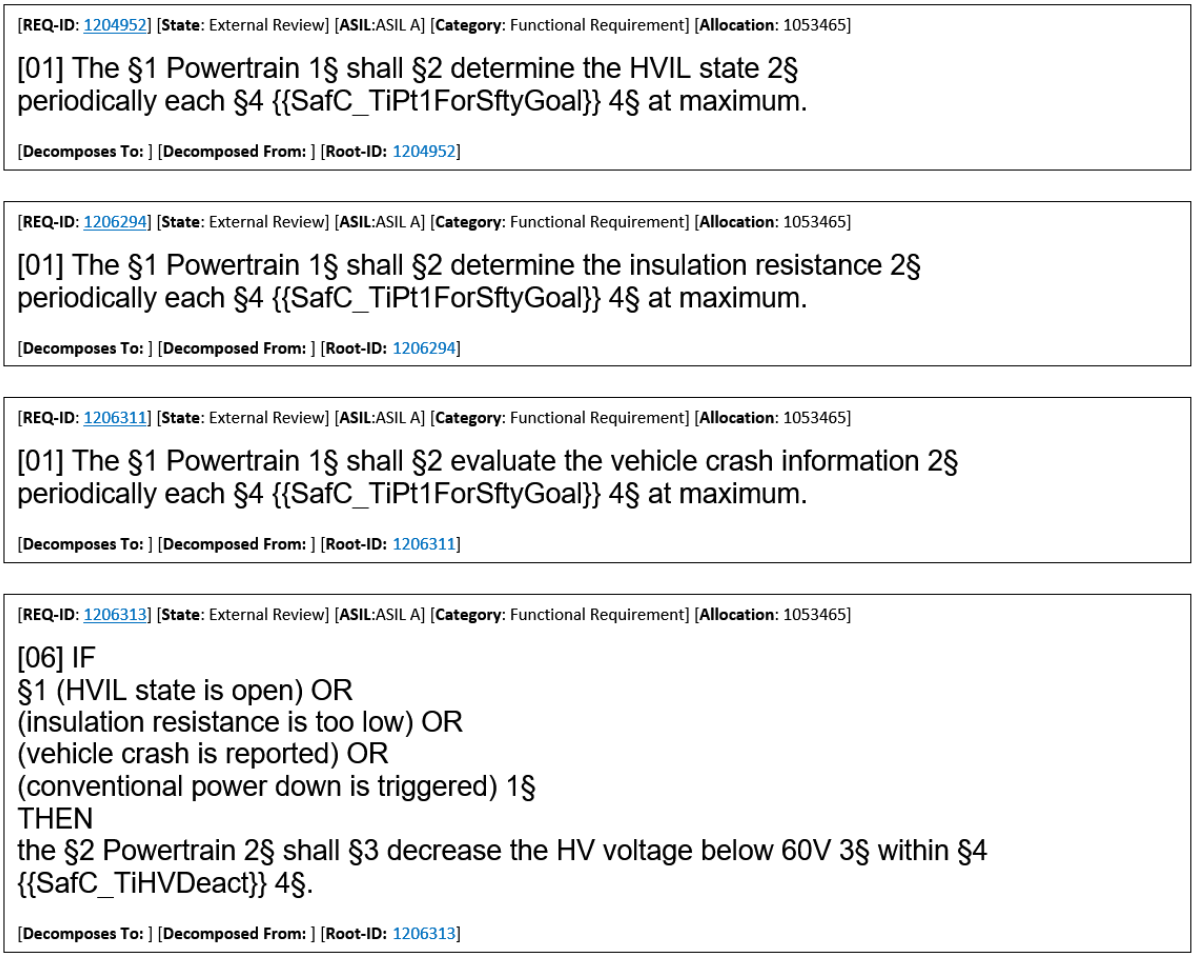

Figure 6.2: Example for functional safety requirements of SG07 [4]

The functional safety requirements on powertrain level are the basis for the requirements on element level and the content on both levels is predefined in the functional safety concept, which was provided by AVL. Due to good cross-section of all types of functional safety requirements, which allows a good insight in the thesis, the HVIL state and interruption, respectively, will be examined as example for the

prevention of the SG07 violation. The other requirements of the powertrain level, will not be examined on element level, because this would be a too extensive example.

The following sections should only give an overview of parts of the developed model. The use of input masks to define e.g. durations are not shown. Furthermore, the functionalities of the blocks are not described in detail, because their description can be found in Section 5.2.

6.1 Overview of the HVIL

The HVIL is, from a physical point of view, a hard-wired electric loop, which passes through several elements, like OBC, HVBATT, DCDC, HVAC and EDRV. The interruption of the loop can be measured and indicates that the high voltage parts of the elements might be accessible, due to e.g. an open cover. Accessible high voltage parts are a risk for humans and this shall be prevented. Therefore, all the elements which are passed through the HVIL are monitored for their accessibility. In case of accessibility are appropriate measures triggered.

This is displayed schematically in Figure 6.3. The connections between the elements, which are blue, represent the hard-wired HVIL. The black arrow represents CAN-Bus information about the HVIL state, which is sent from the HVBATT to the HCU. The red arrows represent the CAN-Bus information of the safety mechanisms, which is sent from the HCU to all high voltage elements to prevent high voltage accidents by demanding a discharge.

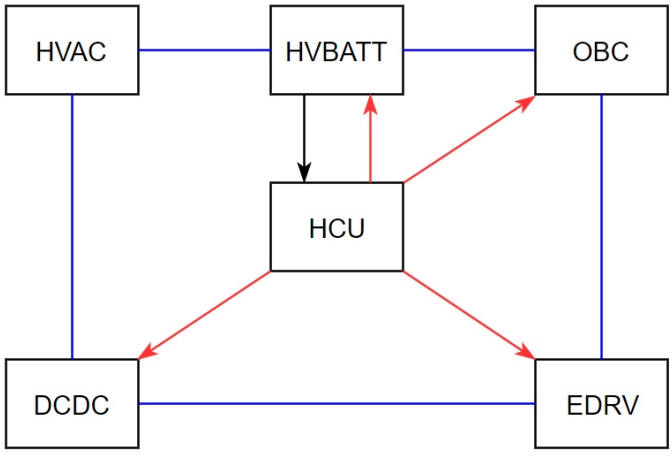

Figure 6.3: HVIL from a physical point of view

For the presented model it is important to mention the simplifications of the modeled HVIL. Although the HVIL is a physical existing hard-wired loop, in this model several elements will communicate the state of the HVIL to the HVBATT, without being in a loop. This was in accordance with the guidelines of AVL and is displayed in Figure 6.4. The coloration corresponds with Figure 6.3.

The HVBATT receives all hard-wired HVIL information and determines if any HVIL signal indicates an open state. If necessary, the state information is set to “open”. This information is received by the

HCU, which crosschecks the signal with the vehicle speed, to determine if the accessible high voltage parts are a risk. In case that the vehicle speed is below a specific value and the HVIL state information indicates an open HVIL, the safe state for the power shut down coordination is triggered and as a result, all high voltage parts are discharged.

The following section will show the described functionality in detail. Figure 6.4 is helpful to follow the descriptions.

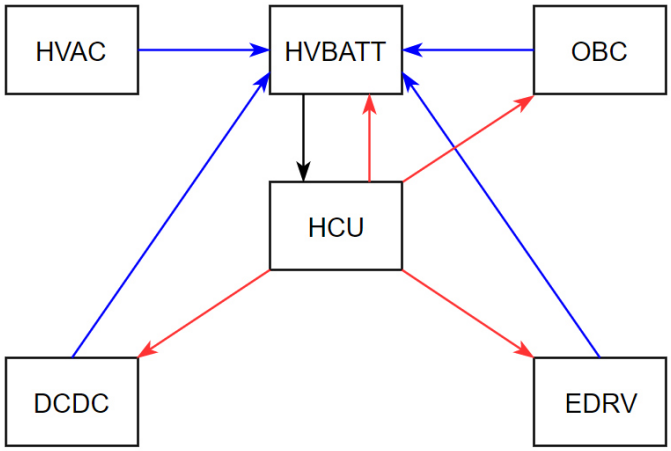

Figure 6.4: HVIL from a model point of view

6.2 Determination of the HVIL Status

The HVIL state is determined in the following elements:

- HVAC
- HVBATT
- OBC
- EDRV
- DCDC

An overview of one of these elements is displayed in the Figure 6.5. Similar models exist for all other elements. In Figure 6.5 one can see the Simulink model of the EDRV with several input and output ports. The “Out_EDRV_HVIL” and the “EDRV CAN output information (E2E protected)” are the output information regarding the hard-wired HVIL and the CAN-Bus information, respectively. This information is sent to at least one other element. The other output ports, which are connected with “Output” blocks, are the representation of the connection to the physical EDRV element. On element level, the element safety mechanisms for e.g. the HVIL are modeled.

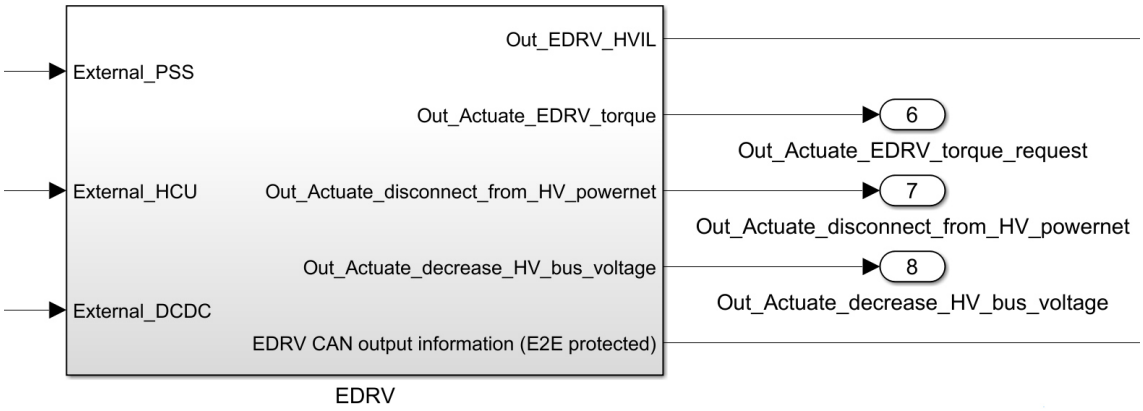

Figure 6.5: Model of the EDRV

The following figure displays a part of the element level of the EDRV. Amongst other ESMs, such as the "ESM[EDRV].206 Vehicle Crash Report Handling", one should pay attention to the EDRV's HVIL ("ESM[EDRV].206 HVIL"). Another remark concerns the "Input" blocks of Figure 6.6, which were seen as ports in Figure 6.5. These blocks receive information from other elements and pass them on to the ESM responsible for the E2E check.

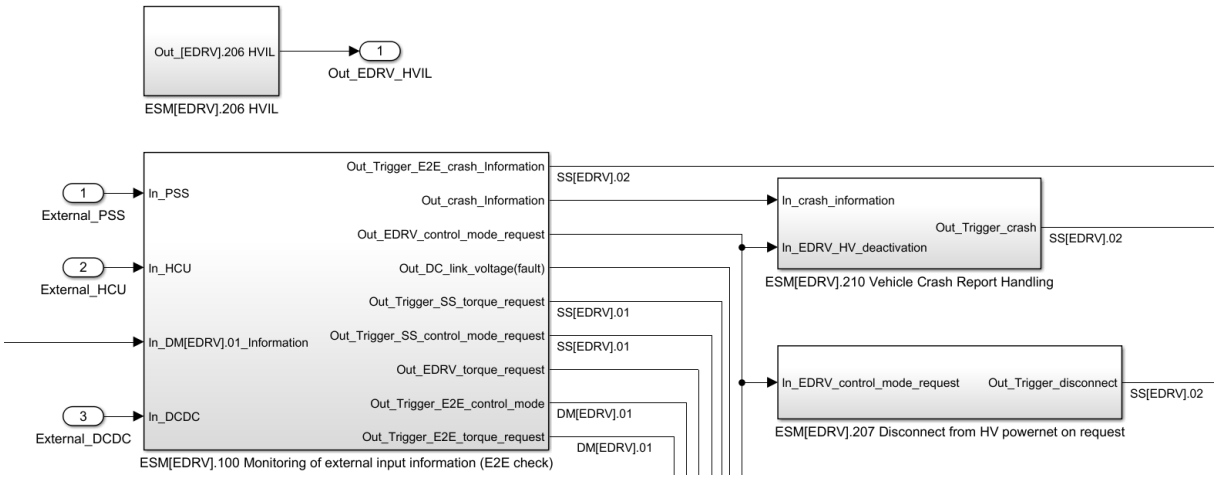

Figure 6.6: Model of the element safety mechanisms

The ESM of the HVIL is described with the functional safety requirement displayed in Figure 6.7. The requirement demands the interruption of the HVIL in case the high voltage parts are accessible. Similar requirements exist for all high voltage elements.

ESM[EDRV].206 HVIL

[**REQ-ID:** 1206365] [**State:** External Review] [**ASIL:**ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060139]

[06] IF
§1 high voltage parts of the EDRV can be accessed 1§
THEN
the §2 EDRV 2§ shall §3 interrupt the HVIL line 3§ within §4 {{SafC_TiEdriveHVIL}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1206365]

Figure 6.7: HVIL requirement [4]

The following two figures show the modeled subsystem of the EDRV's HVIL in Simulink and in Integrity Modeler. The HVIL state is determined according to the functional safety requirement of Figure 6.7. A block from the developed library determines the HVIL state. The "Out_[EDRV].206 HVIL" block passes the determined information on to the element level.

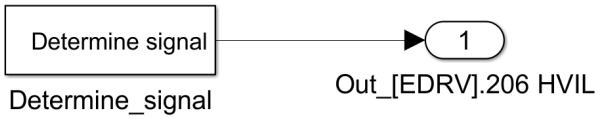

Figure 6.8: Subsystem representing the requirement of the EDRV's HVIL in Simulink

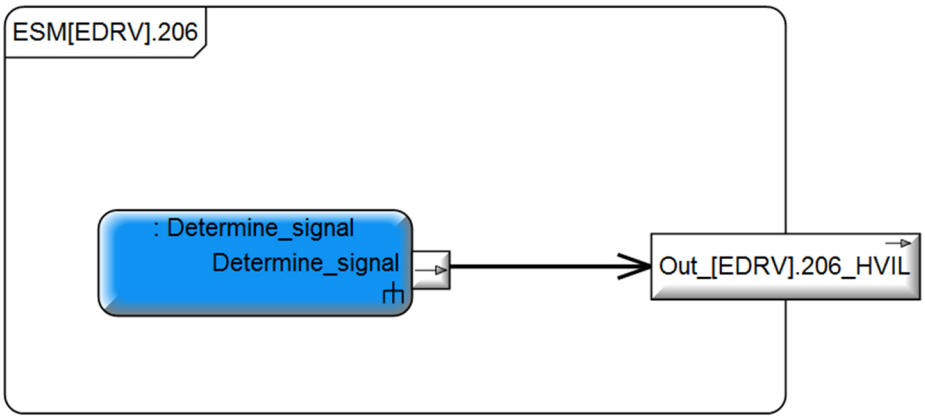

Figure 6.9: Subsystem representing the requirement of the EDRV's HVIL in Integrity Modeler

On the ERDV's element level, the HVIL signal is not used for any other purpose. Hence, the HVIL signal is being sent via an "Output" block to the HVBATT. Other signals of the EDRV will not be tracked in this example.

6.3 Processing of Received Signals

Amongst other signals, the HVBATT receives the EDRV's HVIL signal. In case of the HVIL of one of these signals, or the HVBATT's HVIL, being interrupted, the HVBATT shall set the HVIL state into open. The following figure shows the corresponding functional safety requirements.

ESM[HVBATT].206 HVIL monitoring

[REQ-ID: 1201851] [State: External Review] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation: 1060141]

[01] The §1 HV Battery 1§ shall §2 monitor the HVIL 2§ periodically each §4 {{SafC_TiDetn1CanInpInfo}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1201851]

[REQ-ID: 1427332] [State: In Work] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation:]

[02] IF
§1 (HVIL is interrupted) 1§
THEN
the §2 HVBATT §2 shall §3 set the HVIL state into open 3§ within §4 {SafC_TiFltDetn1}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1427332]

Figure 6.10: Requirements of the HVBATT's HVIL [4]

On the HVBATT's element level this is modeled with the element safety mechanism "ESM[HVBATT].206 HVIL", which is displayed in Figure 6.11.

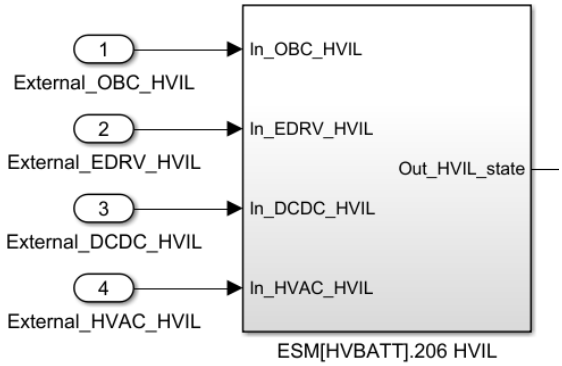

Figure 6.11: HVBATT's element safety mechanism

The model of the ESM, which reflects the functional safety requirements, is shown in the following two figures. Figure 6.12 shows the Simulink model and Figure 6.13 the SysML model. Besides the "Input" blocks for HVILs from other elements, the HVIL state is also determined in the HVBATT. In case one of these HVILs is interrupted, the "OR" block is triggered and as a result, the "Trigger_set_state" block changes the state from "HVIL closed" to "HVIL open".

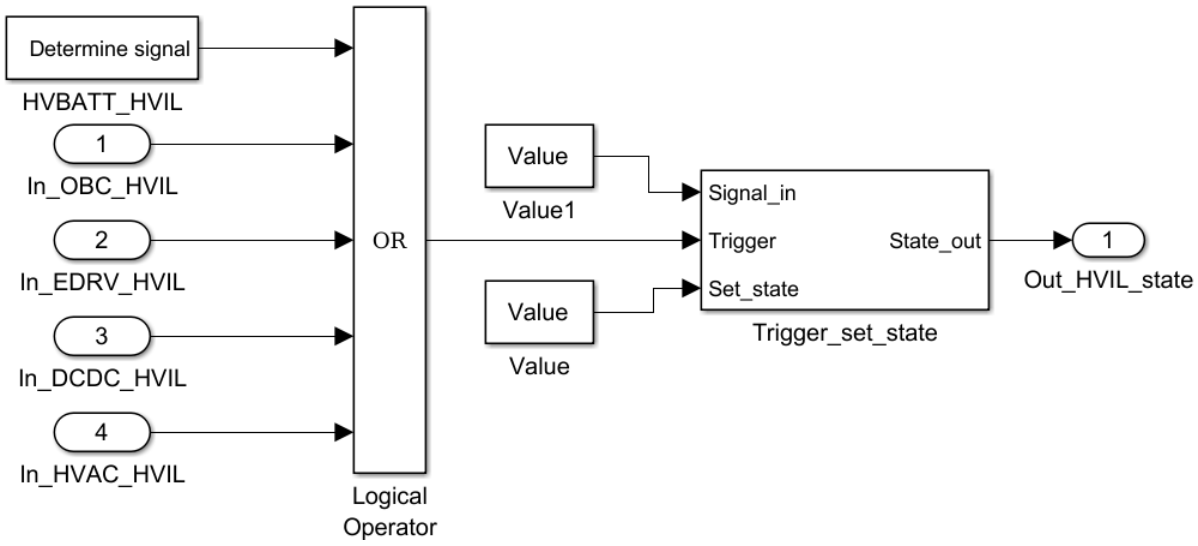

Figure 6.12: Simulink model of ESM[HVBATT].206 HVIL

One difference between the Simulink and the Integrity Modeler model, which should be mentioned, are the ports and pins, respectively. For the Simulink model, every port is used for only one signal. In contrary, the “OR” pin of the Integrity Modeler block is used by various signals.

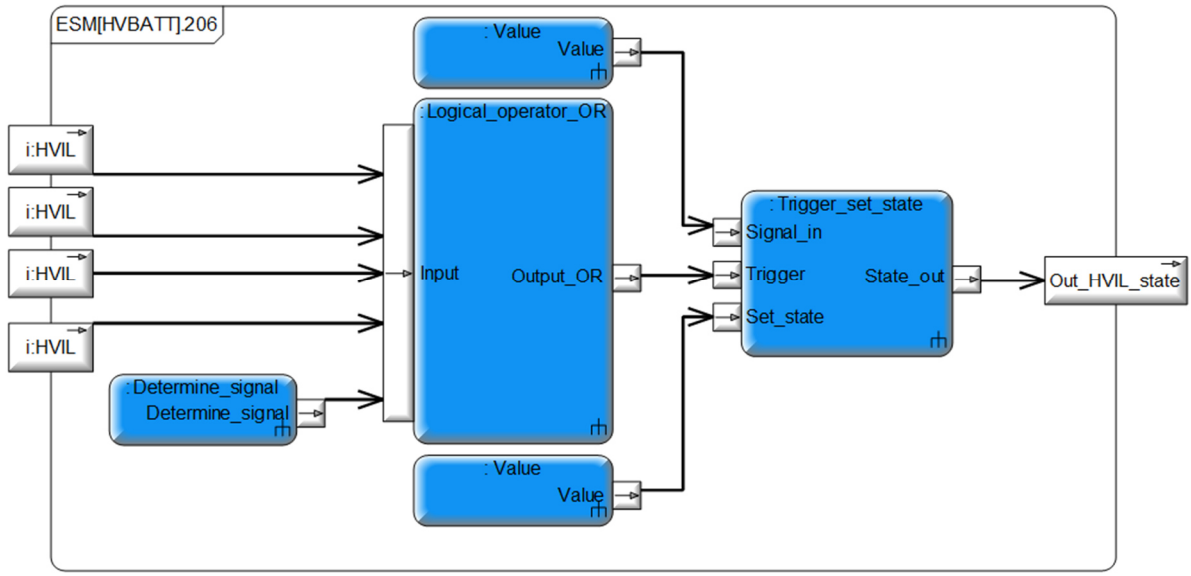

Figure 6.13: SysML model of ESM[HVBATT].206 HVIL

On the HVBATT’s element level, the HVIL state is not used in any other functional safety requirement, except in the E2E protection. Due to the HVIL state being safety relevant, the correct signal transmission is essential. Therefore, the HVIL state is E2E protected. The corresponding functional safety requirement is shown in the following figure. As mentioned in Section 5.3, the E2E protection is modeled in a very simplified way for this thesis.

ESM[HVBATT].300 E2E protection of safety relevant output information

[REQ-ID: 1165385] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060141]

[01] The §1 HV Battery 1§ shall §2 apply end-to-end protection means to safety relevant output information 2§ periodically each §4 {{SafC_TiComctnForHvbat}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1165385]

Figure 6.14: Requirement for E2E protection of the HVIL state [4]

The corresponding model to the functional safety requirement, presented in Figure 6.14, is displayed in Figure 6.15 (Simulink) and Figure 6.16 (SysML). Apart from the HVIL state, the SS[HVBATT].02 information and the insulation state are also E2E protected and afterwards the signals are combined in a “Mux” block and fed to the CAN-Bus.

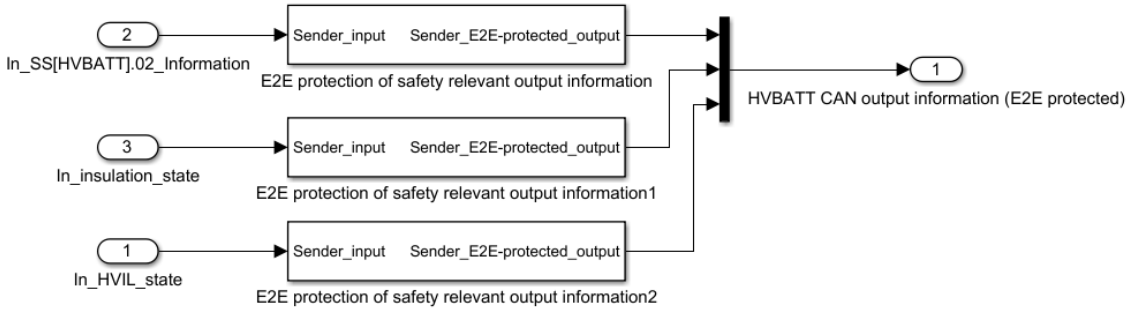

Figure 6.15: Simulink model of E2E protection of the HVIL state

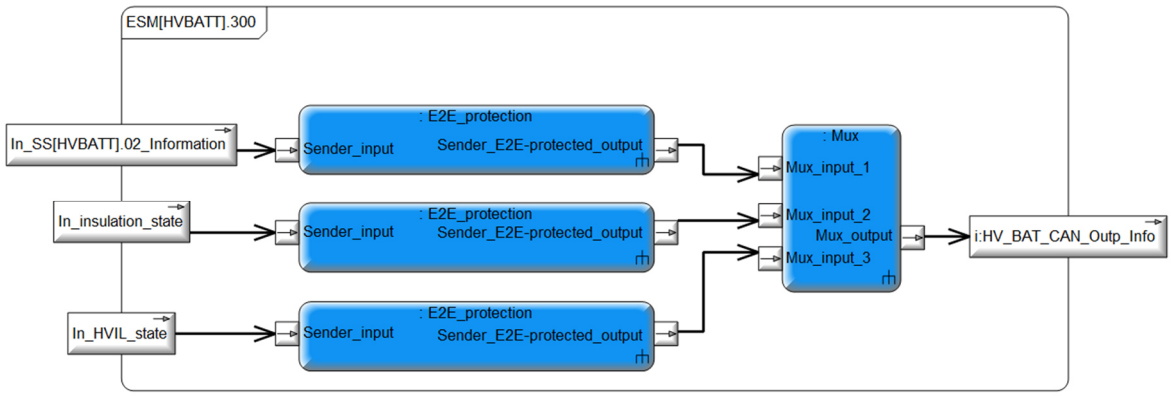

Figure 6.16: SysML model of E2E protection of the HVIL state

The following figure shows an overview of the HVBATT's element level with the involved ESMs to fulfill the HVILs functional safety requirements.

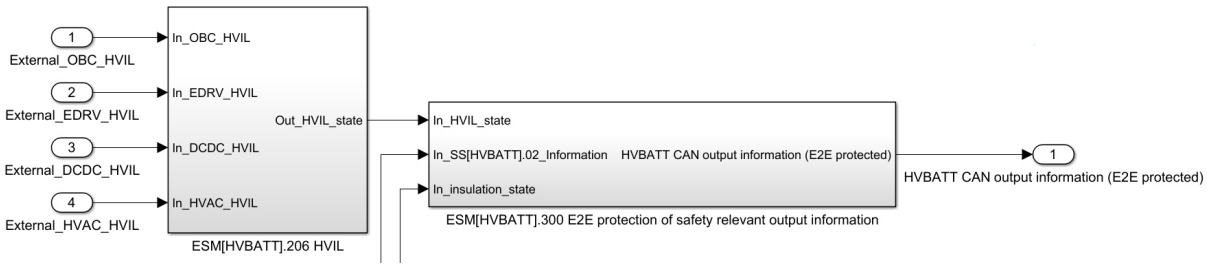

Figure 6.17: HVBATT’s element level for the HVIL

6.4 Trigger of Safety Measures

The HVBATT CAN-Bus information is received by the HCU. Due to safety reasons, the HVIL state is E2E protected. Hence, the HCU must check the correct transmission. This is demanded by the element safety mechanism which is shown in Figure 6.18. This functional safety requirement rules out any communication disturbances.

ESM[HCU].100 Monitoring of external input information (E2E check)

[REQ-ID: 1147337] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060149]

[01] The §1 HCU 1§ shall §2 perform an end-to-end check on all safety relevant input communication signals 2§ periodically each §4 {{SafC_TiDetn1CanInpInfo}} 4§ at maximum.

[Decomposes To:] [Decomposed From:] [Root-ID: 1147337]

[REQ-ID: 1425928] [State: In Work] [ASIL:ASIL A] [Category: Functional Requirement] [Allocation: 1060149]

[02] IF
§1 (an E2E failure of the HVIL state information is detected) 1§
THEN
the §2 HCU 2§ shall trigger §3 SS[HCU].05 3§ within §4 {{SafC_TiFltDetn1}} 4§.

[Decomposes To:] [Decomposed From:] [Root-ID: 1425928]

Figure 6.18: E2E ceck of the HVIL state signal [4]

The model which reflects these functional safety requirements are displayed in the following two figures (depiction in Simulink and SysML). One can see the partitioning of the HVBATT signal with a “Demux” block and that the HVIL signals CRC, range and validity are checked with the appropriate blocks. If one of the corresponding blocks is triggered, an E2E failure signal is prompted and provided to other ESMs. Regardless of a possible E2E failure, the HVIL state is transmitted to other ESMs. A more detailed explanation of signals and E2E failures can be found in Section 5.2.4.

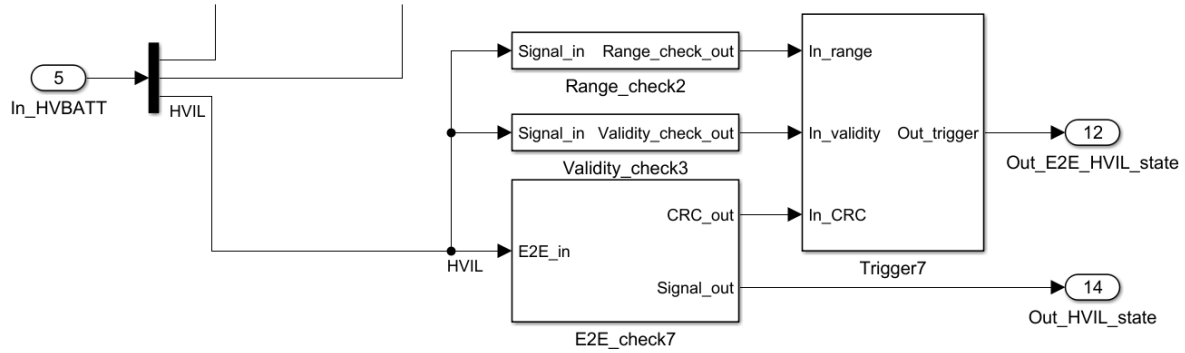

Figure 6.19: Simulink model of the E2E ceck of the HVIL state signal

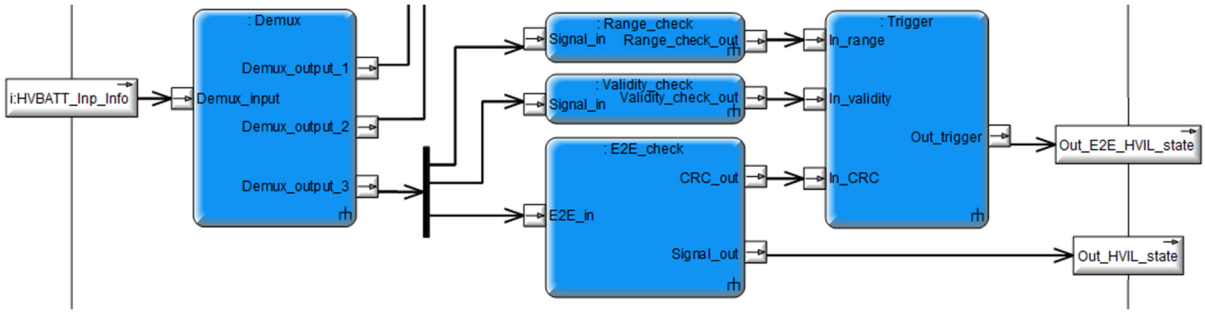

Figure 6.20: SysML model of the E2E ceck of the HVIL state signal

In case of an E2E failure of the information about the HVIL signal being triggered, the functional safety requirement demands the trigger of “SS[HCU].05 Warn driver in case of hybrid powertrain failure”. This safe state will be described later. First, the “HVIL state” signal will be examined.

The “HVIL state” signal is requested by the element safety mechanism “ESM[HCU].214 HVIL fault handling”. The corresponding functional safety requirement is shown in Figure 6.21. Besides the “HVIL state” the requirement also demands the vehicle speed. In case of the HVIL state information indicating an interruption and the vehicle speed being below a specific value, the “SS[HCU].08 HV shut down coordination” is triggered.

ESM[HCU].214 HVIL fault handling

[**REQ-ID:** 1426017] [**State:** In Work] [**ASIL:**ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060149]

[02] IF
§1 (HVIL state information received via the vehicle communication bus indicates a HVIL interruption) AND
(the vehicle speed is below {{SafC_VVehForHvilEvln}}) 1§
THEN
the §2 HCU 2§ shall trigger §3 SS[HCU].08 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1426017]

Figure 6.21: Requirement for the HVIL fault handling [4]

The model for the HVIL fault handling is displayed in Figure 6.22 and 6.23, respectively. The HVIL signal is crosschecked with a predefined value for equality. The vehicle speed is also crosschecked, but for its size. If both crosschecks are triggered, the “And” block is triggered and an information about a faulty HVIL is sent to the “SS[HCU].08”.

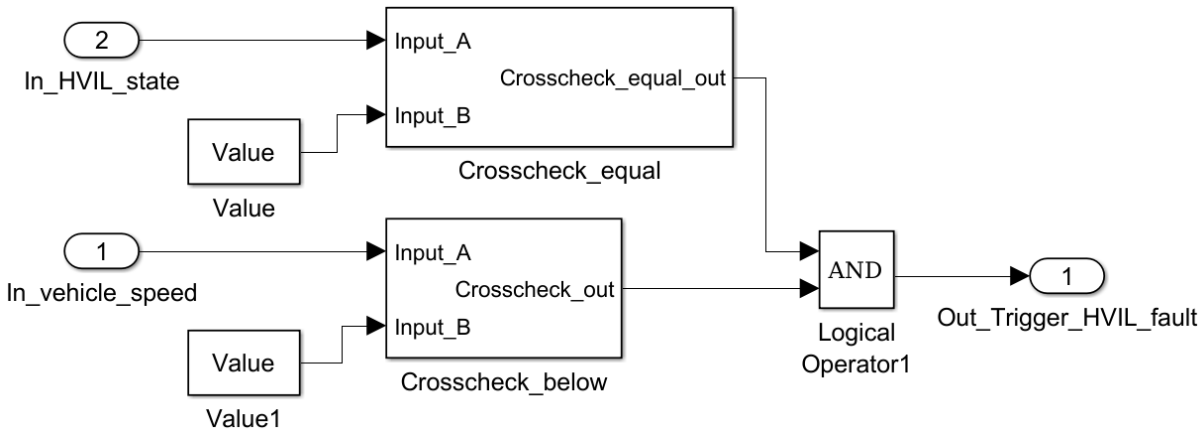

Figure 6.22: Simulink model of the HVIL fault handling

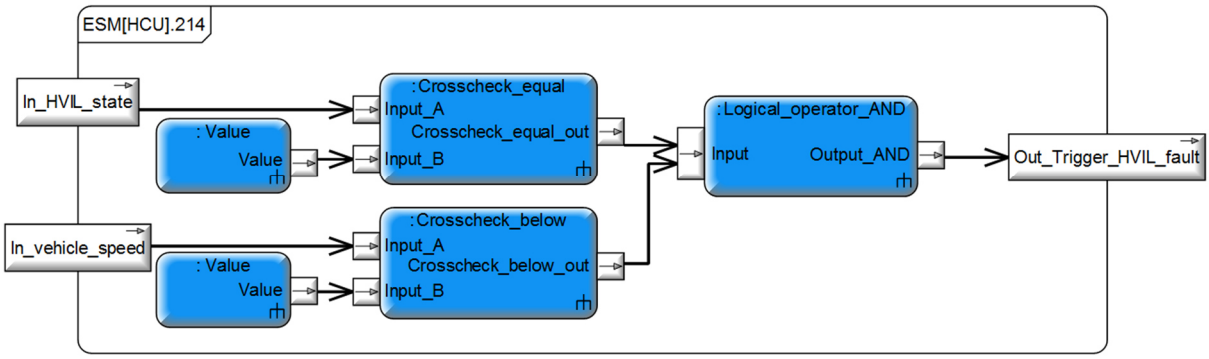

Figure 6.23: SysML model of the HVIL fault handling

The functional safety requirement which describes “SS[HCU].05 Warn driver in case of hybrid powertrain failure”, is shown in the following figure. In case the safe state is triggered, the original signal gets replaced by a signal, which warns the driver via the HMI (Human machine interface) about a hybrid powertrain failure until the vehicle is restarted.

SS[HCU].05 Warn driver in case of Hybrid Powertrain failure

[REQ-ID: 1202237] [State: External Review] [ASIL:ASIL C] [Category: Functional Requirement] [Allocation: 1060149]

[03] IF
§1 SS[HCU].05 1§ is triggered
THEN
§2 HCU 2§ shall §3 provide Hybrid Powertrain failure information to the HMI 3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§ until restart.

[Decomposes To:] [Decomposed From:] [Root-ID: 1202237]

Figure 6.24: Requirement of SS[HCU].05 [4]

The model which describes the safe state is displayed in Figure 5.25 and 5.26. Apart from the described E2E failure signal of the HVIL, several other signals can also trigger the safe state.

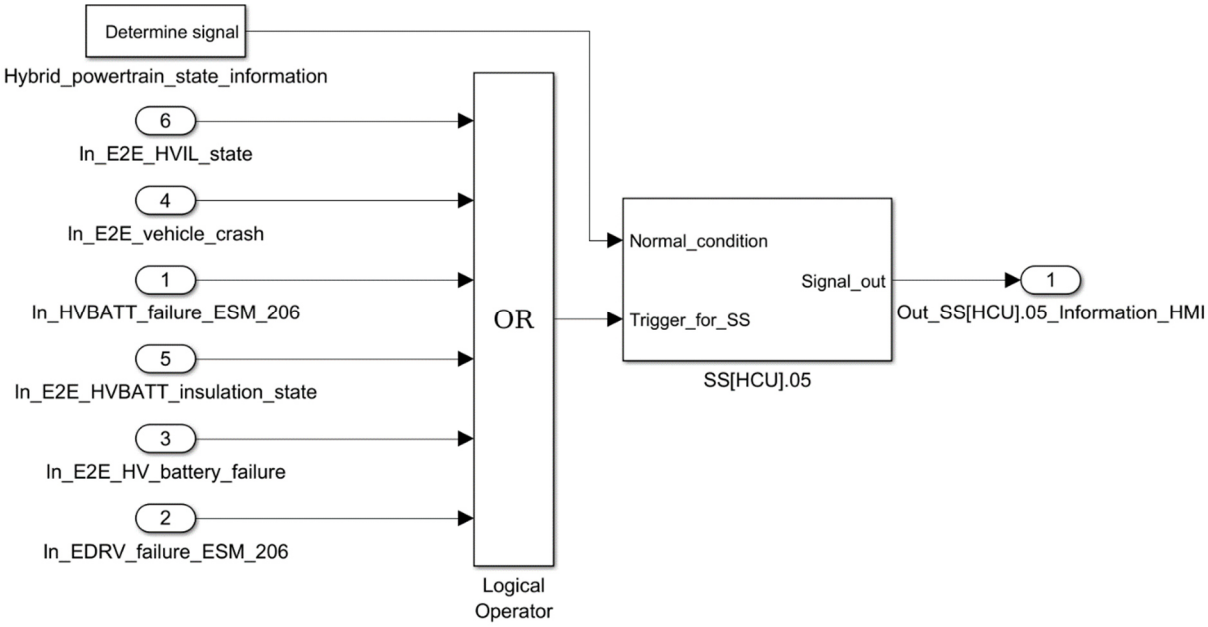

Figure 6.25: Simulink model of SS[HCU].05

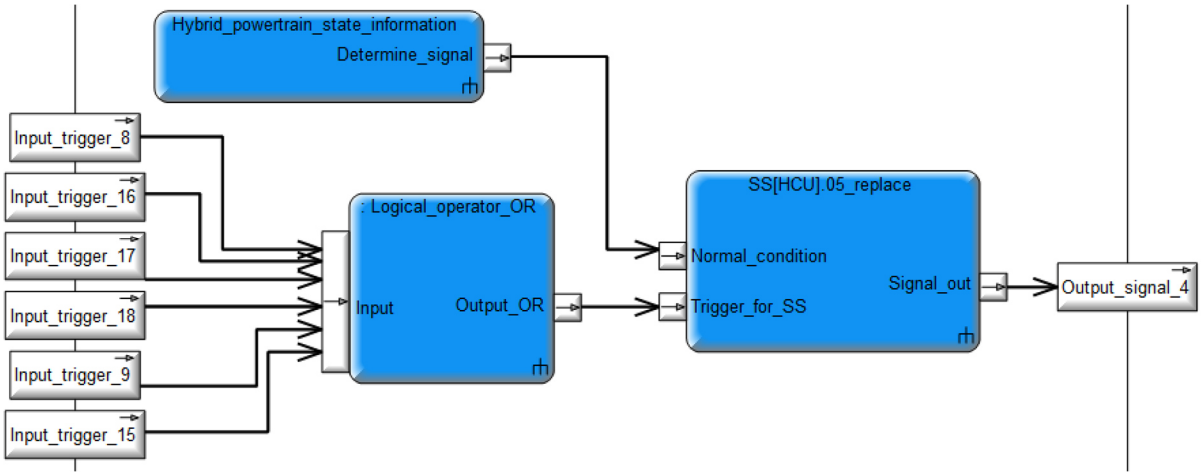

Figure 6.26: SysML model of SS[HCU].05

The safe state requirement “SS[HCU].08 HV shut down coordination”, which is triggered amongst other signals by the HVIL state, is shown in the following figure. It is an unusual safe state, because the functional safety requirement demands the fulfillment of a specific sequence of actions. The developed block library does not offer any suitable block, or block combination. Hence, the unique block is used to model this part of the functional safety requirement. Figure 6.28 shows the corresponding model.

SS[HCU].08 HV shut down coordination

[**REQ-ID:** 1425651] [**State:** In Work] [**ASIL**:ASIL A] [**Category:** Functional Requirement] [**Allocation:** 1060149]

[03] IF
§1 SS[HCU].08 1§ is triggered
THEN
§2 HCU 2§ shall §3 perform the following actions in exactly the following order:
1. Request the EDRV, OBC and DC/DC to disconnect from the HV powernet
2. Wait until the absolute value of the HV Battery current drops below {{SafC_IOpenContHVDeact}} or until {{SafC_TiOpenContTol}} elapse
3. Request the HV Battery to disconnect from the HV powernet
4. Request the EDRV, OBC and DC/DC to perform an active discharge
3§ within §4 {{SafC_TiFltRctn1ForSSt}} 4§ until restart.

[**Decomposes To:**] [**Decomposed From:**] [**Root-ID:** 1425651]

Figure 6.27: Requirement of SS[HCU].08 [4]

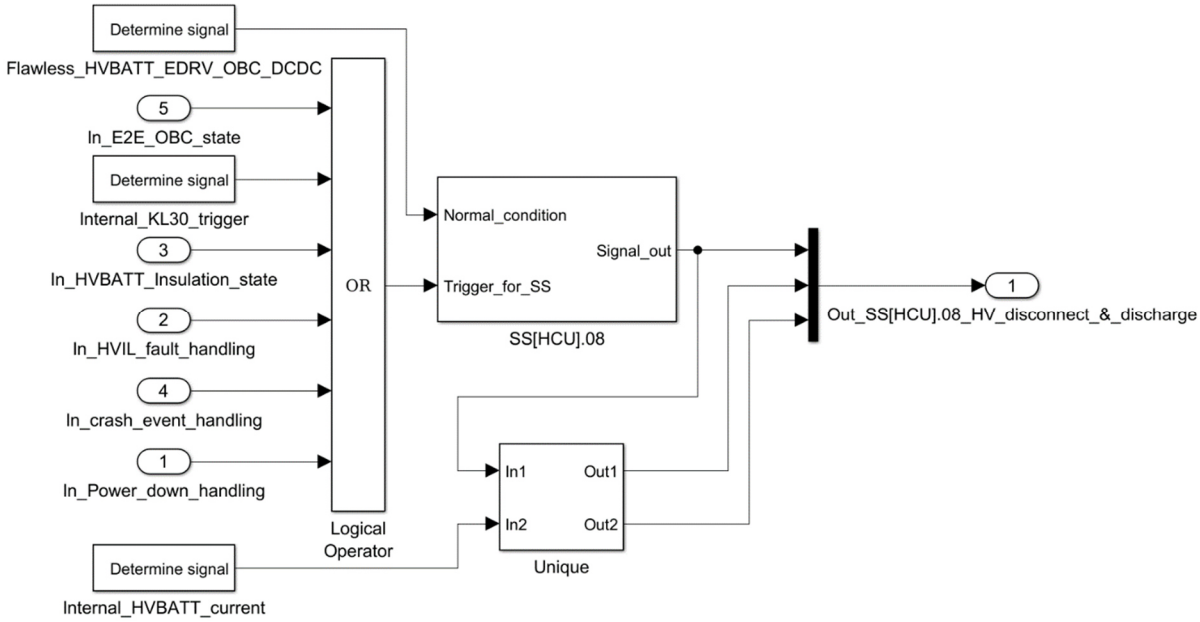

Figure 6.28: Simulink model of SS[HCU].08

The information from the “SS[HCU].08 HV shut down coordination” is safety relevant and therefore E2E protected. This is similar to the E2E protection, shown in Figure 6.15 and 6.16 and will therefore not be presented again. The E2E protected signals are being sent via the CAN-Bus to the relevant elements. The relevant elements carry out an E2E check, similar to the one shown in Figure 6.19 and 6.20. Subsequently, the signals and the potential E2E failures are processed in the corresponding elements, similar to previously described ESMs and the occurrence of a hazardous event is prevented.

7 DISCUSSION

The reviewed literature indicates that model-based approaches are increasingly important to cope with the challenges of modern automotive development. This also includes the domain of functional safety development.

AVL also implements model-based approaches in the functional safety domain, but the manual creation of functional safety concepts is still problematic. The analysis of the current state revealed problems with consistency, reusability, a lack of information clarity, and especially a lack in ambiguity. Moreover, the limitations of natural language are revealed.

The model-based approach presented in this thesis with predefined and reusable blocks is a promising approach. Although the approach is only a foundation for further developments, the blocks allowed the modeling of 95 % of all functional safety requirements of a safety goal with only 27 predefined blocks. The requirements which were modeled with unique blocks indicate the necessity to expand the functionality of the existing blocks, or create blocks which allow adaptations to these requirements. Apart from that indicates the statistical result potential improvements in blocks for E2E protection and crosscheck blocks, because a high percentage of all used blocks are part of these types. A possible improvement for the four different crosscheck blocks could be the combination in one block, where the type of crosscheck can be chosen.

Although the used software tools differ in many ways, both tools fulfill the key aim of this thesis, which is the creation of a reusable library with predefined blocks. A recommendation is not part of the thesis' scope.

Based on the presented example several potential improvements are foreseeable with the presented approach. Due to the future implementation of the approach into the vehicle model and derivation of the functional safety requirements from the model, the consistency of information is improved and the information source is clear. Moreover, the library with predefined blocks and the graphical modeling improves the clarity of functional safety requirements and the communication between engineers of different domains. In addition to that allows the library a less cumbersome reusability.

Overall is the presented model-based approach an improvement compared to the present approach, but further developments are necessary. To get an executable model and text-based requirements, the blocks must be programmed, connected to databases and the structure must be analyzed automatically. Furthermore, the statistical results indicate that at least the crosscheck-blocks and the communication between elements might offer further potential for improvement.

8 CONCLUSION AND FUTURE WORK

This chapter summarizes the main aspects of the thesis, draws a conclusion and gives an outlook on future work.

8.1 Conclusion

Due to the production of increasingly complex vehicles, the automotive industry faces numerous challenges in the development process of modern vehicles. Reasons for the complexity increase are for example HEVs (Hybrid Electric Vehicle) with batteries and electric machines, the overall growing number of E/E systems and the control of all these components with embedded systems. Failures and faults of the embedded systems are hazardous and therefore safety-critical. The development of safety-critical systems is regulated by the ISO 26262 and adds additional difficulties to the development of modern vehicles. This thesis contributes to an improvement.

Investigated publications revealed that the automotive industry faces the challenges of modern automotive development with the change to model-based approaches. Model-based approaches are expected to integrate multiple development domains, allow faster development processes with less faults, improve traceability and allow a better reuse of earlier developments. Although the challenges are known and model-based development is a promising solution, the industry struggles with the realization.

The development at AVL is also being pushed toward model-based approaches. The creation of functional safety requirements for the safety goals of functional safety concepts is still text-based and lacks in automation, which causes several problems. The aim of this thesis was the creation of a foundation to model the functional safety concept, or parts of it, with blocks and take advantage of the model-based approach, to overcome the problems of the current approach.

Based on the analysis of a text-based functional safety concept of an HEV, which was provided by AVL, it was possible to identify recurring patterns in the functional safety requirements. The found patterns refer to the structure and content of functional safety requirements and were identified to be describable with blocks.

These blocks were created in Matlab/Simulink and with SysML in Integrity Modeler and allow specifications via input masks or the block properties, respectively. For reuse, the blocks were gathered in a library, which was created for each of the software tools.

To show the feasibility of this approach, the blocks in the library were used successfully for the modeling of functional safety requirements of a safety goal. The blocks allowed the reproduction of

95 % of all functional safety requirements. The rest was created with a block for unique purposes. The results allow the conclusion that a model-based approach with blocks is an improved solution for the creation of parts of the functional safety concept, but future work is necessary.

8.2 Future Work

The blocks were used successfully to model functional safety requirements, but additional work is necessary. Due to the limitations of these thesis, the created blocks are dummies, which are not executable and do not create any content. Hence, further developments are necessary in these areas to use the blocks in actual functional safety development processes. Moreover, a solution for unique requirements is necessary. This solution might be the expansion of the already used block, offering an input mask and a linkage to a database.

Future work must also investigate the automated creation of the text-based functional safety requirements from a created model. Therefore, a linkage between the blocks and databases, the automated analysis of a created model and the creation of text-based functional safety requirements is missing. Furthermore, the statistical results indicate that at least the crosscheck-blocks and the communication between elements might offer further potential for improvement.

REFERENCES

[1] R. Aarenstrup: "Managing Model-Based Design," Natick, The MathWorks, 2015.

[2] P. O. Antonino and M. Trapp: “Improving Consistency Checks between Safety Concepts and View Based Architecture Design,” in 12th International Probabilistic Safety Assessment and Management Conference, Honolulu, 2014.

[3] AVL List GmbH: “Requirements Engineering of E-Drive Components (RQvsSPEC),” Graz, 2015.

[4] AVL List GmbH: “Functional Safety Concept (A19593),” Graz, 2017.

[5] K. Beckers et al.: “A Structured and Model-Based Hazard Analysis and Risk Assessment Method for Automotive Systems,” in IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), Memphis, 2013, vol. 158, pp. 238–247.

[6] M. Bialy et al.: “A Methodology for the Simplification of Tabular Designs in Model-Based Development,” in IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering, Florence, 2015, pp. 47–53.

[7] M. Broy et al.: “Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments,” Proc. IEEE, [w. loc.], 2010, vol. 98, no. 4, pp. 526–545.

[8] S. Burton and A. Habermann: “Automotive Systems Engineering und Functional Safety: The Way Forward,” in Embedded Real Time Software and Systems, Stuttgart, 2012.

[9] Chen Tao: “Functional Safety Concept Design of Hybrid Electric Vehicle following ISO 26262,” in 2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific), [w. loc.], 2012, no. August 2012, pp. 1–6.

[10] J. D’Ambrosio and G. Soremekun: “Systems Engineering Challenges and MBSE Opportunities for Automotive System Design,” in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, 2017, pp. 2075–2080.

[11] M. Ehsani et al.: "Modern Electric, Hybrid Electric, and Fuel Cell Vehicles," 2nd ed., vol. 6. CRC Press, [w. loc.], 2004.

[12] T. Farkas et al.: “Integration of UML with Simulink into Embedded Software Engineering,” ICCAS-SICE Int. Jt. Conf., pp. 474–479, Fukuoka, 2009.

[13] G. H. Fisher: “Model-based systems engineering of automotive systems,” in 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conference. Proceedings (Cat. No.98CH36267), Bellevue, 1998, vol. 1, p. B15/1-B15/7.

[14] A. B. Fotso and A. Rettberg: “State of the art for mechatronic design concepts,” in Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, Suzhou, 2012, pp. 232–240.

[15] J. Friedman: “MATLAB/Simulink for Automotive Systems Design,” in Proceedings of the Design Automation & Test in Europe Conference, Munich, 2006, pp. 1–2.

[16] H. Giese et al.: “Model Synchronization at Work: Keeping SysML and AUTOSAR Models Consistent,” in Graph Transformations and Model-Driven Engineering, G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, Eds. Springer-Verlag Berlin Heidelberg, 2010, pp. 555–579.

[17] M. Hillenbrand: “Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen,”, Karlsruher Institut für Technologie, Doctoral Thesis, 2012.

[18] J. Holtmann et al.: “A Seamless Model-Based Development Process for Automotive Systems,” Softw. Eng., pp. 79–88, [w. loc.], 2011.

[19] H. Kraus: “Development of an Operation Strategy for Plug-in Hybrid Electric Vehicles: Long-term Prediction and Adaptation based on Past Vehicle and Driver Data Graz University of Technology,” Graz, Graz University of Technology, Doctoral Thesis, 2016.

[20] G. Lami and F. Falcini: “Automotive SPICE Assessments in Safety-Critical Contexts: An Experience Report,” in 2014 IEEE International Symposium on Software Reliability Engineering Workshops, Naples, 2014, pp. 497–502.

[21] G. Macher: “Framework for the Integrated Model-Based Development of Dependable Automotive Systems and Software,” Graz University of Technology, Doctoral Thesis, 2015.

[22] R. Mader: “Computer-Aided Model-Based Safety Engineering of Automotive Systems,” Graz University of Technology, Doctoral Thesis, 2012.

[23] R. Mader et al.: “A Bridge from System to Software Development for Safety-Critical Automotive Embedded Systems,” in 2012 38th Euromicro Conference on Software Engineering and Advanced Applications, Cesme, 2012, pp. 75–79.

[24] R. Mader et al.: “A Framework for Model-Based Safety Requirements Round-Trip Engineering,” in 10th IET System Safety and Cyber-Security Conference 2015, Bristol 2015, pp. 1–6.

[25] H. Martin et al.: “Challenges for Reuse in a Safety-Critical Context: A State-of-Practice Study,” in SAE Technical Paper 2014-01-0218, Detroit, 2014, p. 12.

[26] H. Martin et al.: “Model-based Engineering Workflow for Automotive Safety Concepts,” in SAE Technical Paper, [w. loc.], 2015.

[27] A. Meroth et al.: “Optimization of the development process of intelligent transportation systems using automotive SPICE and ISO 26262,” in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp. 1481–1486.

[28] A. Rauzy and C. Bleriot-Fabre: “Model-Based Safety Assessment: Rational and trends,” in 2014 10th France-Japan/ 8th Europe-Asia Congress on Mecatronics (MECATRONICS2014), Tokyo, 2014, pp. 1–10.

[29] B. Sari and H.-C. Reuss: “A model-driven approach for the development of safety-critical functions using modified architecture description language (ADL),” in 2016 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC), Toulouse, 2016, pp. 1–5.

[30] J. Schäuffele and T. Zurawka: "Automotive Software Engineering," 6th ed., Wiesbaden, Springer Fachmedien Wiesbaden, 2016.

[31] V. Socci: “Implementing a model-based design and test workflow,” in 2015 IEEE International Symposium on Systems Engineering (ISSE), Rome, 2015, pp. 130–134.

[32] H. Sporer: “Mechatronic System Development: An Automotive Industry Approach for Small Teams,” Graz University of Technology, Doctoral Thesis, 2016.

[33] Y. Vanderperren and W. Dehaene: “From UML/SysML to Matlab/Simulink: Current State and Future Perspectives,” in Proceedings of the Design Automation & Test in Europe Conference, Munich, 2006, pp. 1–1.

[34] D. D. Ward: “System safety in hybrid and electric vehicles,” in ASSC ’11 Proceedings of the Australian System Safety Conference - Volume 133, Melbourne, 2011, pp. 79–84.

[35] T. Weilkiens: "Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur," 3. Edition. Heidelberg, dpunkt.verlag, 2014.

[36] P. Wilson and H. A. Mantooth: "Model-based engineering for complex electronic systems: techniques, methods and applications," Waltham, Elsevier Inc., 2013.

[37] Won Hyun Oh et al.: “Model-Based Development of Automotive Embedded Systems: A Case of Continuously Variable Transmission (CVT),” in 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA’05), Hong Kong, 2005, pp. 201–204.

[38] [Online], The Mathworks Inc.: Available: https://de.mathworks.com/help/simulink/ug/creating-block-libraries.html#bu3dj9s-1. [Accessed: 20-Dec-2017], “Create a Custom Library.”

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AVL	Anstalt für Verbrennungskraftmaschinen List
ASIL	Automotive Safety Integrity Level
AUTOSAR	Automotive Open System Architecture
BMS	Battery Management System
C0	Separation Clutch
CAN-Bus	Control Area Network - Bus
CCU	Clutch Control Unit
CRC	Cyclic Redundancy Check
CVT	Continuously Variable Transmission
DCDC	Direct Current – Direct current
DM	Degradation Mode
E/E	Electric / Electronic
E-Drive	Electric-Drive
E2E	End to End
EAST-ADL	Electronics Architecture and Software Technology – Arch. Description Language
ECU	Electronic Control Unit
EDRV	Electric-Drive
EMS	Engine management system
ESP	Electronic Stability Program
ESM	Engine Management System
FMEA	Failure Mode and Effects Analysis
FTA	Fault Tree Analysis
GSN	Goal Structure Notation
GUI	Graphical User Interface
HARA	Hazard analysis and risk assessment
HCU	Hybrid Control Unit
HEV	Hybrid Electric Vehicle
HMI	Human Machine Interface
HV	High voltage
HVAC	Heating, Ventilation and Air Conditioning
HVBATT	High Voltage Battery
HVIL	High Voltage Interlock Loop

IBD	Internal Block Diagram
ICE	Internal Combustion Engine
ICM	Instrument Cluster (HMI Dashboard)
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
KL15	Clamp 15
KL15 (HW)	Clamp 15 (Hardwired)
LV	Low Voltage
MCU	Motor control unit
OBC	On-Board Charger
OEM	Original Equipment Manufacturer
OMG	Object Management Group
PEPS	Passive Entry Passive Start
PSS	Passive Safety System
PTC	Parametric Technology Corporation
PTSM	Powertrain Safety Mechanism
Px-HEV	Parallel-x Hybrid Electric Vehicle
QM	Quality Management
RBS	Regenerative Braking System
REQ-ID	Requirement- Identifier
SRF	Safety Relevant Function
SGxx	Safety Goal (xx = Number of Safety Goal)
SS	Safe State
SysML	Systems Modeling Language
TCU	Transmission Control Unit
TQ	Torque
UML	Unified Modeling Language

A-APPENDIX