TU WIEN Informatics

Model-driven methods for developing ISO-certified safety-critical systems

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Laurenz Gutleder, BSc
Matrikelnummer 01228773

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Mag. Dr. Christian Huemer
Mitwirkung: Dipl.-Ing.in Mag.a Dr.in Alexandra Mazak-Huemer

Wien, 5. Mai 2020

Laurenz Gutleder	Christian Huemer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

TU WIEN Informatics

Model-driven methods for developing ISO-certified safety-critical systems

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Laurenz Gutleder, BSc
Registration Number 01228773

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Mag. Dr. Christian Huemer
Assistance: Dipl.-Ing.in Mag.a Dr.in Alexandra Mazak-Huemer

Vienna, 5th May, 2020

Laurenz Gutleder	Christian Huemer

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Laurenz Gutleder, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 5. Mai 2020

Laurenz Gutleder

Danksagung

Mein besonderer Dank gilt meinen BetreuerInnen Prof. Dr. Christian Huemer und Dr.in Alexandra Mazak-Huemer, die mich während des gesamten Diplomarbeitsprozesses bestmöglich unterstützt haben. Von der Themenfindung bis hin zur finalen Korrektur standen sie mir immer hilfreich sowie professionell zur Seite.

Des Weiteren möchte ich mich bei Dr. Konrad Wieland bedanken, der mit seinem Fachwissen und seiner Erfahrung aufschlussreiche Ideen in diese Arbeit einbrachte. Außerdem danke ich Sabine Wolny und Bernhard Wally für ihre Ratschläge und die Weitergabe ihrer wissenschaftlichen Expertise vor, aber auch während meiner Arbeit.

Ein großer Dank gebührt auch meiner Partnerin Sandra für ihre Unterstützung während meines Studiums sowie für das Lektorat dieser Diplomarbeit. Während der Entstehung dieser Arbeit war sie eine besonders geduldige und motivierende Kraft in meinem Leben.

Natürlich möchte ich mich auch von ganzem Herzen bei meinen Eltern Susanne und Karl für ihre großartige und unentwegte Unterstützung während meiner gesamten Ausbildung bedanken. Nur durch ihren Rückhalt wurden mir mein Studium und somit auch diese Arbeit ermöglicht.

Abstract

Since the release of the IEC 61508 international standard for Functional Safety of Electrical/Electronic/Programmable Electronic (E/E/PE) Safety-related Systems and its specific variants, classical methods of system development have quickly reached their limits due to sophisticated safety requirements such as full traceability. One possible approach to address the issue of managing the ever increasing complexity in the development processes of certified safety-critical systems is model-based systems engineering (MBSE). Different model-based methods are applied depending on their safety-critical domains and specific safety standards such as the ISO 26262 for functional safety of road vehicles rather than applying a set of general methods based only on the parent standard IEC 61508.

The first part of the work comprises a Systematic Mapping Study (SMS) investigating and classifying a high number of scientific publications to analyze the similarities and differences of the applied model-based methods and their reasons of application in the standards IEC 61508, ISO 26262 for road vehicles, EN 50128 for railway systems, and DO-178 for airborne systems. Based on the results, a Systematic Map is created in order to identify patterns of model-based methods in different sectors of industry.

The second part of the thesis addresses the question of an appropriate means to represent the ISO 26262 Automotive Safety Integrity Level (ASIL) tailoring and decomposition concept in a SysML extension. This includes the development of a SysML profile, a use case, and a descriptive evaluation based on the requirements specified in the ISO 26262 series of standards. This SysML profile can help to simplify the modeling and decomposition requirements for the Automotive Safety Integrity Level concept.

Kurzfassung

Seit der Veröffentlichung der internationalen Normenserie IEC 61508 für die funktionale Sicherheit elektrischer/elektronischer/programmierbarer elektronischer (E/E/PE) Systeme, die eine Sicherheitsfunktion ausführen, und ihrer anwendungsspezifischen Varianten sind die klassischen Methoden der Systementwicklung aufgrund anspruchsvoller Sicherheitsanforderungen wie der vollständigen Rückverfolgbarkeit an ihre Grenzen gestoßen. Ein möglicher Ansatz zur Bewältigung der immer komplexer werdenden Entwicklungsprozesse von zertifizierten sicherheitskritischen Systemen ist die modellbasierte Systementwicklung. Je nach dem sicherheitskritischen Anwendungs- und Geltungsbereich und der zu erfüllenden Sicherheitsnorm, zum Beispiel die ISO-Norm 26262 für die funktionale Sicherheit von Kraftfahrzeugen, werden unterschiedliche spezifische modellbasierte Methoden angewandt, anstatt eine Reihe allgemeiner Methoden anzuwenden, die nur auf der übergeordneten Norm IEC 61508 basieren.

Der erste Teil der Arbeit umfasst eine Systematic Mapping Study (SMS), in der eine große Anzahl wissenschaftlicher Publikationen untersucht und klassifiziert wird, um die Gemeinsamkeiten und Unterschiede der angewandten modellbasierten Methoden und die Gründe für ihre Anwendung in den Normen IEC 61508, ISO 26262 für Kraftfahrzeuge, EN 50128 für Eisenbahnsysteme und DO-178 für Avionik-Systeme zu analysieren. Auf der Grundlage der Ergebnisse wird eine Systematic Map erstellt, um Muster modellbasierter Methoden in verschiedenen Industriesparten zu identifizieren.

Der zweite Teil der Arbeit befasst sich mit der Frage nach einem geeigneten Mittel zur Darstellung des in der ISO-Norm 26262 spezifizierten Automotive Safety Integrity Level (ASIL) Zuweisungs- und Zerlegungskonzepts in einer SysML-Erweiterung. Dies beinhaltet die Entwicklung eines SysML-Profils, eines Anwendungsfalles und einer ausführlichen deskriptiven Evaluierung auf der Grundlage der in der ISO 26262 Normenreihe festgelegten Bedingungen. Das entwickelte SysML-Profil kann dazu beitragen, die Modellierungs- und ASIL-Zerlegungsanforderungen für das Konzept des Automotive Safety Integrity Level zu vereinfachen.

Contents

CHAPTER 1

Introduction

This chapter is dedicated to providing an introduction and comprehensive overview of the contents and conducted research of this thesis. Firstly, the motivational reasons for this thesis are described. Secondly, the problem statement is framed to derive accurate research goals for this thesis. Afterward, the aim of the work is explained to specify which activities are part or are not part of this thesis, followed by the methodological approach to determine how this research is conducted. The last section describes the structure of the work.

1.1 Motivation

Today, technological progress is increasingly making its way into all areas of our workaday life. One of these areas is safety-critical systems engineering, which is confronted with many development challenges due to this rapid technical progress. Safety-critical systems can be defined as systems that have the potential to create safety-related issues in case they do not operate according to their clear stipulation or do not work as designed [Lev95, Eri05]. In order to allow the correct definition of safety activities during the life cycle development of the systems, safety-critical systems are usually analyzed using systematic safety processes [Bah97]. These systematic safety processes are mostly determined in the corresponding safety standards. One of these standards is, for instance, the IEC 61508 international standard for functional safety of electrical/electronic/programmable electronic (E/E/PE) safety-related systems. In the past, these safety analyses were usually performed rather late in the safety cycle because of limited possibilities such as simulation. This led to a very high expenditure of time and costs, which threatens to grow exponentially with the ever increasing efforts in safety-critical systems development. For instance, the requirements engineering dilemma states that "failures can best be found late in the development process but ideally would be found early, already in

the requirements analysis phase. Because then, required safety measures and resulting effort and cost can be planned from the start" [Foc16, p. 12]. Unfortunately, classical approaches have reached their limits due to sophisticated safety requirements such as full-traceability. They are no longer suitable for managing the increasing complexity in the development processes of certified safety-critical systems. Therefore, new systems engineering approaches have become necessary to manage the development of complex safety-critical systems such as advanced automotive driver-assistance systems.

One possible approach to address the issue of increasing complexity is model-based systems engineering (MBSE) or model-driven systems engineering (MDSE). In an MBSE environment, a model is the basis and information source for all development data used or created. This model is usually stored in a central model repository and shared in real-time with all model users, which is the main difference to classical development approaches. In classical approaches documents are created and exchanged between the stakeholders, which results in two fundamental problems for classical approaches. Firstly, it could lead to inconsistencies, and secondly, no single source of information is available and total traceability can hardly be achieved. Apart from that, model-based engineering has many other advantages such as automatic code generation or test case generation [Alt14a, BFK10]. Another useful application example is the support for proof of compliance to a specific functional safety standard [Gal14, LBEK15]. Furthermore, domain-specific modeling languages such as SysML are customized to specifically support systems engineering activities. In summary, the applications of MBSE are manifold and promising [RFB12].

The manifold applications of MBSE for the development of safety-critical systems are the focus of this thesis. Certain model-based methods are recommended or particularly suitable for systems development that complies with a certain safety standard. The motivation for this thesis is to firstly structure model-based methods according to safety-critical domains, functional safety standards, and reasons for using MBSE, and secondly to investigate how a specific model-based method can be created to ensure that the development complies with a part or clause of a functional safety standard. In the next section 1.2, the problem statement is defined based on this motivation.

1.2 Problem Statement

In the previous section 1.1, the manifold applications of MBSE to develop safety-critical systems have been mentioned. The IEC 61508 standard, entitled *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems*, is a functional safety standard applicable to all kinds of industry. More than ten industry- and application-specific variants based on IEC 61508 have been released. For instance, EN 50128[1] for

[1] See [EN5].

railway systems, DO-178[2] for airborne systems, and ISO 26262[3] for road vehicles. These industry- and application-specific variants inherited most of the crucial concepts of IEC 61508. Each of these functional safety standards also has additional extensive requirements on its own. Thus, automated verification of compliance of the developed system artifacts with these standards based on evidence is a major goal in the safety-critical systems engineering field of work [PWSB11]. This goal can be accomplished by using MBSE. Some of the introduced functional safety standards already include model-based approaches in their specifications. For instance, ISO 26262 contains several explicit provisions for using MBSE in contrast to IEC 61508 which contains none [BFK10]. This is the reason why the harmonization of MBSE in the context of IEC 61508 and its industry- and application-specific variants is difficult. Different model-based methods are applied depending on their safety-critical domains and specific standards rather than applying a set of general methods based only on the parent standard IEC 61508. Furthermore, many scientific publications address different model-based approaches in safety-critical industry domains pursuing narrow research goals for specific standards rather than research goals across different industry domains or standards. Thus, a crucial problem is that there is no structure and no big picture of the application of MBSE in the development of functional safety systems—a problem that is to be solved in this thesis by means of a Systematic Mapping Study to structure this field of interest.

The lack of a big picture to structure the application of MBSE in the context of IEC 61508 and its industry- and application-specific variants is not the only problem to be addressed in this thesis. Systems engineering in the context of IEC 61508 and its specific variants is time-consuming and cost-intensive. The ISO 26262 standard, entitled *Road vehicles – Functional safety*, is based on the IEC 61508 standard. Since the release of ISO 26262, document-centric approaches have had issues managing their key requirements such as traceability. The challenging task of managing systems engineering in compliance with ISO 26262 can be supported or accomplished by using MBSE [Adl14, Alt14a, Ros14, FGH+18, LKB19]. This resulted in a growing acceptance of MBSE in the automotive industry [BFK10]. Even though the use of SysML for the development of automotive systems is widespread, SysML cannot represent all concepts in a model as specified by ISO 26262 [Foc16]. Thus, it is necessary to extend existing modeling languages or create new domain-specific languages. Previous domain-specific research in the context of ISO 26262 has focused on the language extension of UML rather than SysML [BHFH13, BCF+14, BCF+16]. There is a lack of research that addresses the problem of how to extend SysML to facilitate ISO 26262 compliant MBSE. The ISO 26262 series of standards comprises twelve parts and more than a hundred requirements. In order to carry out valuable research, this thesis can only investigate a specific part of the ISO 26262 concepts. More precisely, the focus is on the concepts of ASIL tailoring and decomposition. In ISO 26262, every hazardous event determines a corresponding automotive security integrity level (ASIL), ASIL A is the lowest level and ASIL D is the

[2] See [DO-b].
[3] See [ISOd].

highest level. Requirements decomposition with respect to ASIL tailoring is a method to decompose safety requirements into redundant safety requirements to allow ASIL tailoring to the next level of detail. Decomposition is important because, under certain circumstances, the ASIL can be lowered through this technique [ISOm]. Unfortunately there is no option to model this decomposition yet. The question of an appropriate way to represent the ISO 26262 ASIL decomposition concept in a modeling language has not yet been solved. Further details on how the stated problems are approached are given in the next section 1.3.

1.3 Aim of the Work

The aim of the work to approach the problems stated in the previous section 1.2 can be structured into two parts:

1. The first research part focuses on the missing big picture to structure MBSE in the context of functional safety systems engineering and its corresponding underlying standards. Firstly, it is examined in which of the industry- and application-specific variants of IEC 61508 MBSE methods are most frequently used. This leads to the first research question:

   RQ 1: In which safety-critical domains based on/related to IEC 61508 are model-based methods applied? Which standards for functional safety are applied?

   The three most common functional safety standards variants are selected for further analyses. Secondly, the applied MBSE methods and the reasons for applying MBSE methods are surveyed in the context of IEC 61508 and its three industry- and application-specific variants that have been selected based on the conditions in the first step. This leads to the second and third research question:

   RQ 2: What are the reasons for applying model-based methods in IEC 61508 and each of the three most common safety-critical standards in RQ 1?

   RQ 3: Which model-based methods are applied in IEC 61508 and each of the top three most common safety-critical standards in RQ 1?

   In summary, this part of the work aims at structuring this field of work to create a big picture to identify research gaps and encourage further research.

2. The second research part focuses on the modeling of the ASIL decomposition concept specified in [ISOm]. The solution is based on an extension specifically tailored for SysML because it is the state-of-the-art modeling language for ISO 26262 compliant

MBSE [DS10]. This research part examines the feasibility of the following research question:

RQ4: What is an appropriate means to represent the ISO 26262 ASIL decomposition concept in a SysML extension?

The aim of this part of the work is not just the systematic development of the SysML extension but also a detailed evaluation to ensure compliance with the relevant parts and clauses of ISO 26262.

The methodological approach to realize the two research parts comprises a Systematic Mapping Study for the first part and design-science research for the second part. Both methodologies are described in detail in the next section 1.4.

1.4 Methodological Approach

The methodological approach for achieving the expected results consists of the following two parts:[4]

1. **Systematic Mapping Study**:

   The software engineering systematic map, invented by Petersen et al. [PFMM08, PVK15], is a method for building a classification scheme and structure for a software engineering field of interest. This method has been applied in this thesis and includes a combination of different self-defined facets of the scheme to answer more specific research questions. The following steps have been performed (slightly adapted from [PFMM08]):

   (i) Definition of research questions.

   (ii) Conducting of searches in scientific databases (IEEE, Scopus, Springer Link, DBLP, DOMA), on Google Scholar, and in printed guidelines & books about functional safety.

   (iii) Screening of publications to create a result set of relevant publications. The filter criteria are derived from an interview with a domain expert.

   (iv) Classification of publications from the result set of relevant publications based on keywording. The keywording process considers abstracts and conclusions of scientific publications and relevant chapters of guidelines & books.

   (v) Data extraction and systematic mapping process. The systematic mapping results are visualized in charts.

[4] Both parts of the methodological approach are independent of each other.

2. **Design-science research:**

   For the second part of the thesis, a design-science research framework is applied in order to define an appropriate means to represent the ISO 26262 ASIL decomposition concept in a SysML profile. More precisely, this part of the thesis uses the framework for executing and evaluating information systems research combining behavioral-science and design-science paradigms introduced by Hevner et al. [HRM+04]. The framework has been adapted for our purposes and comprises four main aspects:

   (i) Environment: The environment consists of the companies that already apply ISO 26262 compliant model-based systems engineering and their technologies. Enterprise Architect, Model Driven Generation (MDG), and SysML are the chosen technologies and part of the environment.

   (ii) Knowledge base: The Systematic Mapping Study conducted in the first part of the thesis is the foundation of the knowledge base for the development and evaluation of the artifact. The knowledge base is complemented by model-driven methodologies and ASIL definitions and related decomposition rules. Furthermore, it is necessary to include the ISO 26262 series of standards to ensure a complete specification of a use case and evaluation.

   (iii) Development: The development process itself is carried out according to the systematic development approach introduced by Strembeck and Zdun [SZ09].

   (iv) Evaluation: The SysML profile is evaluated by performing a descriptive evaluation. The goal is to build informed arguments to prove the developed artifact's utility.

1.5 Structure of the Work

This thesis comprises five chapters and is structured as follows:

Chapter 2 presents two main components of this work. Firstly, a detailed overview and the state of the art of the model engineering area of expertise are provided. This includes information about the possibilities to create new modeling languages or extend existing ones. Secondly, the international standards for functional safety which are relevant for this thesis are introduced. More precisely, the IEC 61508, ISO 26262, EN 50128, and DO-178 standards are presented. Furthermore, these four safety standards are examined to assess the extent to which model-based engineering is already included in their specifications.

Chapter 3 starts with an introduction about the two research methods Systematic Mapping Study (SMS) and Systematic Literature Review (SLR). Afterward, an SMS is conducted to answer three specific research questions about model-based approaches used for systems engineering in compliance with international standards for functional safety.

Each of the five steps performed during the SMS is explained in detail. The results of the SMS are discussed and visualized in charts. Furthermore, the most interesting publications from the final result set of relevant papers are introduced to provide more information about the basis on which the research questions have been answered.

Chapter 4 describes the systematic development of a SysML extension and domain-specific language (DSL) to define an appropriate means for modeling ISO 26262 compliant ASIL decomposition. At the beginning of the chapter, related work, the ninth part of ISO 26262[5], and the theoretical approach are discussed. The practical approach then inludes the definition of the DSL's core language model, the definition of the behavior of the DSL language element, the definition of the DSL's concrete syntax, and the integration with the DSL's target platform. Afterward, a use case is presented and modeled using the developed DSL followed by a descriptive evaluation of the DSL's model-engineering utility in compliance with the ISO 26262 series of standards.

The final chapter 5 is dedicated to the conclusion of this thesis.

[5] See [ISOm].

CHAPTER 2

Background and State of the Art

This chapter is intended to provide a detailed overview of the background and state of the art of the model engineering area of expertise, including its possibilities to create own modeling languages or extend existing ones. Furthermore, international standards for functional safety which are relevant for this thesis are outlined and serve as an introduction to how safety standards have influenced the paradigm shift from document-centric approaches to model-based approaches. The paradigm shift has happened due to the limited possibilities of document-centric approaches [BFK10]. A detailed discussion about model-based methods and reasons for applying model-based approaches in the presented safety standards can be found in chapter 3.

2.1 Model-Driven Engineering

The history of model-driven engineering (MDE) started back in the 1980s when its predecessor called computer-aided software engineering (CASE) emerged. The goal of CASE was to raise the level of abstraction used to develop software by synthesizing the implementation artifacts from graphical representations. Examples for these graphical representations are state machines or data flow diagrams. This led to essential advantages such as reduced manual coding effort, debugging, and porting programs. In theory, CASE had promising potential and was also well received by the research community. However, it was never widely adopted. Serious problems such as the unsophisticated mapping between the general-purpose graphical language representations and the underlying domain-specific platforms made it difficult to use CASE for developing complex projects [Dou06].

The MDE technology started to emerge with the announcement of the model-driven architecture (MDA) proposal[1] in November 2000 by the Object Management Group [JBB09].

[1] See [MDA].

The MDE is a broader concept than the MDA and described as seen in [DGWC14] as "using models to support automated analysis of a design, as the basis for automatic test generation, and as a source for automatically-generated documentation". Thus, the goals of MDE are to raise the level of abstraction and increase the level of automation. The key aspect of this development approach is that models drive the development process and are the primary artifacts rather than peripheral ones [AGK+14]. The MDE comprises at least two components. Firstly, any kind of system is represented by a model. Secondly, a meta-model defines the abstract syntax to define a model or class of models respectively. This is necessary for defining homologous models and enabling automatic processing of models [MSAA13]. In Figure 2.1, the relation between these two components is illustrated. A model represents a system or a part of a system and at the same time conforms to a meta-model.

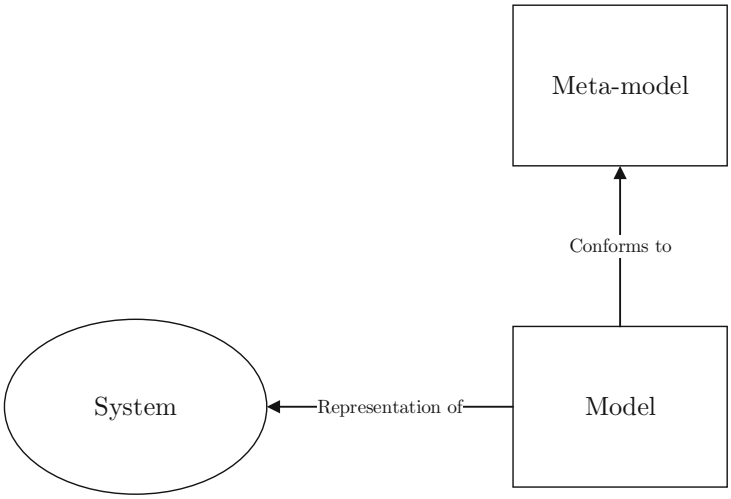

Figure 2.1: Basic relation of representation and conformance in MDE [MSAA13, p. 28].

The MDE technology includes many more possibilities and features such as the definition and design of conceptual model editors, but this section is only intended to give a brief introduction to the background and basics. Further possibilities and features are listed for instance in [BCH13, p. 253f]. In subsection 2.1.1, the model-driven architecture is introduced. In subsection 2.1.2, the history and particular characteristics of model-based systems engineering are explained. In the last subsection 2.1.3, the possibilities to extend existing modeling languages are discussed.

2.1.1 Model-Driven Architecture

The MDA is a comprehensive approach for applying MDE practices to systems development and has been defined by the OMG. Unlike CASE, MDA is widely accepted in the industry and currently the most known modeling framework [BCW17]. The basic

concepts of MDA describe the terms system, model, modeling language, architecture, view & viewpoint, abstraction, architectural layers, transformation, and platform [OMGa, p. 5-10].[2]

In MDA, three modeling levels are distinguished to define the level of architectural abstraction of models, more precisely the computation-independent model, platform-independent model, and platform-specific model [BCW17].
The computation-independent model layer models "real things" rather than representations of "real things" in an information system [OMGa, p. 8]. The platform-independent model describes the behavior and structure in a model without considering the implementation platform. The platform-specific model is usually tied to a particular platform and must contain all required information regarding the behavior and structure. It can be seen as a detailed technology-aware and platform-aware specification of the system [BCW17, OMGa]. Rather than modeling the platform-specific model from scratch, the platform-independent model is usually combined with other information to produce the platform-specific model. This is achieved using model transformations [MB15]. The transformation process is shown in Figure 2.2.

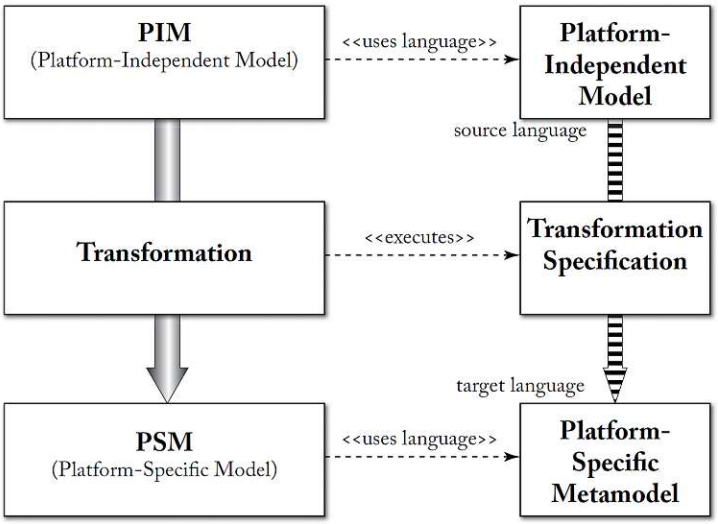

Figure 2.2: Model transformation process to produce the platform-specific model from a platform-independent model [BCW17, p. 48].

The advantages of using MDA models are that they make it easier to deal with the complexity of large systems and they improve the interaction and collaboration between organizations, people, hardware, and software. This is because they define the structure, semantics, and notation of models using industry standards [OMGa].

[2] An explanation for these terms is briefly given in [MB15].

The MDA comprises a family of standards rather than just one single standard. Several well-known standards are part of the MDA, for instance the UML, BPMN, and SysML. Additionally, the Meta Object Facility (MOF) that standardizes the model environment is also part of the MDA family of standards [OMGa]. In summary, the MDA is the foundation of the most state-of-the-art modeling languages.

2.1.2 Model-Based Systems Engineering

Model-based systems engineering (MBSE) is a systems engineering methodology that has gained in importance since the OMG's International Council on Systems Engineering (INCOSE) published its *Systems Engineering Vision 2020*[3] in September 2007. MBSE is defined by INCOSE as follows:

> *"Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software"* [oSEI07, p. 15].

The major goal of the application of MBSE is to replace document-centric approaches widely used for systems engineering with model-based approaches. MBSE has significant advantages compared to document-centric approaches, such as more understandable design change impacts, improved communication of design intents, enhanced analysis of a system design before it is built, and traceability support [Har15].

Several domain-specific modeling languages have been released for MBSE. The language recommended by INCOSE is SysML [oSEI07]. As seen in [BB19, p. 51], "SysML supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities." SysML comprises nine diagram types, more precisely one requirement, four structure, and four behavior diagrams [Gro19a]. More information about the technical aspects of SysML can be found in section 4.3. A more widespread approach to systems modeling is the Object Process Methodology (OPM). The difference to SysML is that in OPM only one diagram emerges. Thus, the creation of several diagrams is dispensable and it can increase consistency and reduce complexity [Dor16]. OPM is publicly available and specified in ISO/PAS 19450:2015 [ISOp].

In summary, MBSE can be used in two ways in systems development. Firstly, as a documentation tool to document legacy systems in order to describe their structure, behavior, and requirements with the means of a modeling language. And secondly, which is the better way, as a powerful systems engineering methodology to enable incremental, iterative, and parallel development life cycles [Alt12, Har15].

[3] See [oSEI07].

2.1.3 Modeling Profiles and Extensions

In chapter 1, the need for model-based approaches to address the issue of managing the ever increasing complexity in systems engineering due to rapid technological progress such as self-driving cars has been discussed. Since these technologies are under constant change, modeling languages should continuously evolve to adapt to the changing needs of the field of work. The creation of new modeling languages is usually achieved by either defining meta-models from scratch, extending the UML, or extending a modeling language that extends the UML itself [LWWC12]. The pros and cons of each option are not discussed in this thesis but more information can be found in [BH]. The modeling language extension possibilities are introduced hereinafter:

- **Meta-model extension**: The meta-model is extended to adapt it to the changing needs of the domain it represents. This is a time-consuming and complex adaption process because it cannot be performed according to the book and requires very specific knowledge of the meta-model to avoid causing problems. Moreover, the extension must also be considered in the modeling environment [LWWC12].

- **UML profile**: The UML profile extension approach is used to extend the UML to customize it for domain-specific purposes. It is not necessary to create a new modeling language as it is done in the meta-model extension approach. Thus, adaptions and extensions can be done easier and faster, and they are less error-prone. The *UML Profiles* packages allow customization in terms of *constraints*, *stereotypes*, and *tagged values*. However, it is not possible to change the semantics of existing UML elements, which is a disadvantage [FV04].

- **Profile for a modeling language already defined as UML profile**: Several state-of-the-art modeling languages are defined as a UML profile, such as SysML[4] and the *Modeling and Analysis of Real-Time and Embedded systems* (MARTE) modeling language[5]. These modeling languages can be extended by defining a separate profile based on the *UML Profiles* package. The procedures used for the extension are exactly the same as for the definition of the modeling language itself. For instance, in [SCS11] MARTE has been extended to incorporate missing security concepts into the MARTE modeling language.

Besides these options, research is trying to combine the best aspects of the meta-model extension and UML profile approaches. For instance, Langer et al. [LWWC12] proposed a solution that adapts the UML profile approach to extend existing modeling languages based on a domain-specific meta-model. However, the perfect universally applicable solution has not yet been found.
In this thesis, a SysML profile is used to extend the SysML. The SysML is defined as a

[4] See [Gro19a].
[5] See [Gro19b].

UML 2 profile and therefore it is also possible to extend SysML elements with the help of the profile mechanism [Alt12, p. 62].

2.2 International Standards for Functional Safety

In 1976 an industrial accident, the Seveso disaster, occurred in a chemical manufacturing plant in the north of Milan in Italy. An unsupervised system reaction caused a high exposure of 2,3,7,8-tetrachlorodibenzo-p-dioxin (TCDD), which resulted in serious damage to humans, wildlife and nature. At that time safety regulations were non-existent and automatic cooling and warning systems were not installed in the factory. As a result of this accident functional safety standards have been specified [Sch10]. The most generic one ever released is the IEC 61508, entitled *"Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems.* Before we focus the international standards in more detail, we will focus on functional safety and functional safety engineering. The International Electrotechnical Commission defines functional safety as:

> *"part of the overall safety relating to the EUC*[6] *and the EUC control system which depends on the correct functioning of the E/E/PE safety-related systems, other technology safety-related systems, and external risk reduction facilities* [IECe, p. 11]"

Thus, functional safety engineering involves the identification of threatening failures and then the setting of maximum tolerable frequency targets for each mode of failure. Every equipment which could contribute to the identified failures must be referred to as "safety-related" and, therefore, must be part of the safety engineering processes. Safety engineering processes include safety functions, which are referred to as a piece of equipment. Safety functions are responsible for maintaining a safe state, detecting hazardous events and transforming failures to a safe state. It is important to note that there are two types of failures in the context of safety-critical systems in the E/E/PE systems domain. Firstly, there are random hardware failures that can be quantified and assessed in terms of failure rates. Secondly, there are systematic failures that cannot be quantified in terms of failure rates. Due to the fact that systematic failures cannot be quantified, the concept of integrity levels was introduced to categorize design techniques and operating techniques by levels. A Safety Integrity Level (SIL) is set based on the maximum tolerable failure rate. The maximum tolerable failure rate must be evaluated for each hazard and depending upon its relative contribution to each hazard. Safety Integrity Levels provide numerical targets to be met for each level, such as the average probability of failure on demand per year (PFDavg). This concept is used in the IEC 61508 functional safety standard and in all safety standards based on the IEC 61508 standard in the same, adapted or specialized approach [SS16].

[6] End-user computing

The IEC 61508 is a generic functional safety standard and applicable to almost all kinds of E/E/PE systems industries. Nevertheless, there are several industry- and application-specific variants for functional safety such as the ISO 26262 for functional safety of road vehicles. A complete overview of the IEC 61508 industrial standard for functional safety and its specific industry sector variants is represented in Figure 2.3. Standards marked with a red square are discussed in detail in the following sections 2.2.1-2.2.4. These four specific standards were selected because the results of the Systematic Mapping Study, described in chapter 3, showed that they are the most relevant ones for investigating the research questions of this thesis.

2.2.1 IEC 61508 - Functional Safety of E/E/PE Safety-related Systems

The International Electrotechnical Commission (IEC) is a worldwide organization founded in 1906 for standardization comprising all national electrotechnical committees and IEC national committees. Its objective is to promote international cooperation on all questions concerning standardization in the electrical and electronic fields. The IEC 61508 standard entitled *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems* was first released by the IEC in 1997 and last updated in 2010. The underlying fundamental concept is that any safety-related system must work correctly or fail in a predictable and safe way when one or more of such systems incorporate electrical/electronical/programmable electronic devices [IECb].

The IEC 61508 consists of the following seven parts:

- Part 1 - *General requirements:* Specifies the general requirements that are applicable to all parts of the standard and can be summarized as the overall framework for the achievement of functional safety. Primarily, it describes why SIL targets need to be specified and how to design, operate and maintain functional safety and manage the hierarchy of documents. IEC 61508:1[7] defines four Safety Integrity Levels, SIL 1 being the minimum level and SIL 4 being the highest level [IECb]. According to [SS16], SIL 1 still requires good design practice and the work effort should not be underestimated in the system's safety life cycle. Furthermore, SIL 4 is the most burdensome level to achieve and is usually avoided due to the need for state-of-the-art safety engineering techniques.

- Part 2 - *Requirements for electrical/electronic/programmable electronic (E/E/PE) safety-related systems:* Covers the safety system hardware and overall system hardware design and applies to all safety-related systems which contain at least one E/E/PE based component. Furthermore, it applies to all subsystems and the components of these systems. IEC 61508:2 specifies how to refine the information

[7] The parts of the standard are written with a subsequent colon and numbered in ascending order.

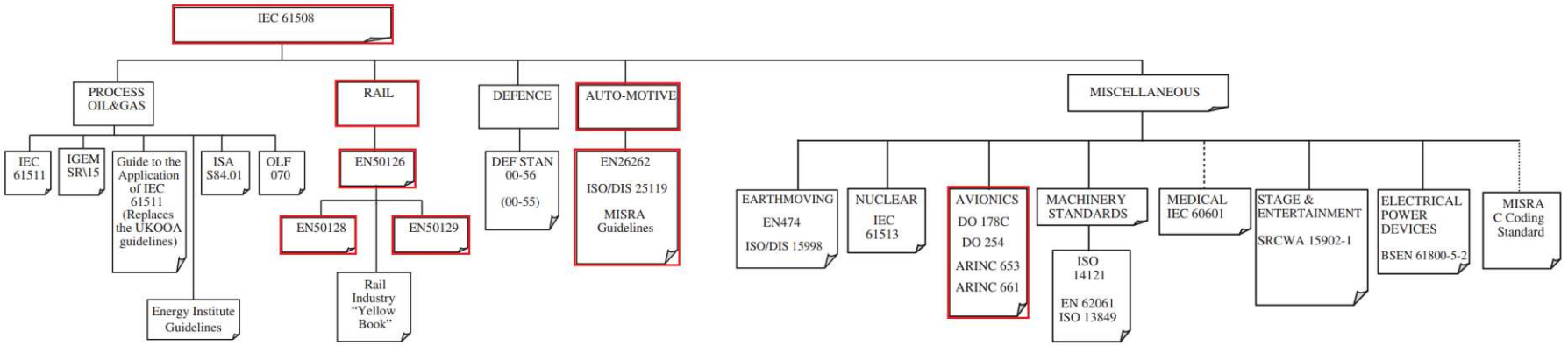

Figure 2.3: IEC 61508 and its industry-specific/application-specific variants (adapted from [SS16, p. 15]).

developed in the first part, concerning safety requirements and their allocation, and how to refine overall safety requirements into safety function requirements and safety integrity requirements. Furthermore, necessary activities, techniques, measures, and more information about requirements engineering are defined [IECc].

- Part 3 - *Software requirements:* Specifies the development of safety-related software and applies to any developed software part of a safety-related system or software used to develop a safety-related system. IEC 61508:3 introduces the important software safety integrity and development life cylce model as shown in Figure 2.4, also frequently quoted as "V-model" [Cla09, IECd]. This model is one of the main reasons for applying model-driven engineering approaches. Even though there is a higher effort of time and costs on the left side responsible for the decomposition and definition of the model, overall results showed a significant reduction of time and costs on the bottom and right side of the model when applying model-based engineering approaches [Kir11].

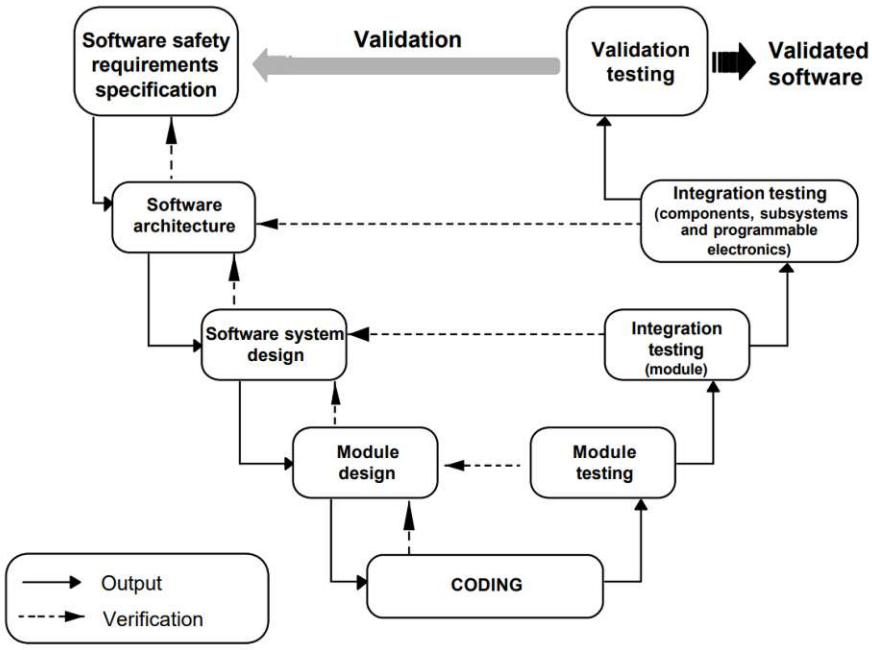

Figure 2.4: Software safety integrity and the development life cycle, also known as the "V-model" (adapted from [IECd, p. 13]).

- Part 4 - *Definitions and abbreviations:* Lists all definitions, abbreviations and explanations used in parts 1 to 7. For instance, the terms fault, fault avoidance, fault tolerance, failure, and random failure are specified in order to keep the scope of interpretation as small as possible. This part is important to understand the specific descriptions in all other parts [IECe].

- Part 5 - *Examples of methods for the determination of safety integrity levels:* Provides information about the underlying concepts of risk and the relationship of risk and safety integrity. Furthermore, IEC61508:5 presents methods for determining safety integrity levels for E/E/PE safety-related systems. Both quantitative and qualitative methods are described. The quantitative method is based on formulas including several numerical parameters. Two examples of qualitative methods are the hazardous event severity matrix and the risk graph [IECf].
- Part 6 - *Guidelines on the application of parts 2 and 3:* Outlines examples and example techniques for calculating the probabilities of failure, calculating diagnostic coverage, methodologies for quantifying common cause failures, and application examples for software safety integrity tables [IECg]. As the calculations are very complex, examples are of great importance.
- Part 7 - *Overview of techniques and measures:* Contains an overview of various safety techniques and measures relevant to part 2 and part 3. For instance, analog signal monitoring is a possible technique to control random hardware failures of electronic components. On the software side, for example, the Vienna Development Method (VIM) can be used to support achieving software safety integrity [IECh].

In this section, the IEC 61508 standard has been introduced and its specific seven parts have been discussed in detail. Most concepts such as the hazard and risk analysis or the probabilistic analysis for determining safety integrity levels are transferred, adopted or specialized in the industry- and application-specific variants of the generic parent standard IEC 61508. Nevertheless, our Systematic Mapping Study (SMS) in chapter 3 shows that model-based methods are mostly applied dependent on their safety-critical domains and application-specific standards rather than applying a set of general methods based only on the parent standard IEC 61508. Thus, a detailed comparison would be an objective for future work.

2.2.2 ISO 26262 - Road Vehicles – Functional Safety

The International Organization for Standardization (ISO) is an independent, non-governmental international organization based in Geneva, Switzerland, with a membership of 164 national standards bodies [ISOo]. The ISO 26262 standard for functional safety of electrical and/or electronic systems in production automobiles entitled *Road vehicles - Functional Safety* is based on the IEC 61508 standard and was first released in 2011. ISO 26262 specifies a functional safety life cycle for automotive products and applies to all activities during the safety life cycle of safety-related systems comprised of electrical components, electronic components, and software components. The release from 2011 consists of the ten parts [ISOd]:

- Part 1 - *Vocabulary*
- Part 2 - *Management of functional safety*

- Part 3 - *Concept phase*
- Part 4 - *Product development at the system level*
- Part 5 - *Product development at the hardware level*
- Part 6 - *Product development at the software level*
- Part 7 - *Production and operation*
- Part 8 - *Supporting processes*
- Part 9 - *Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses*
- Part 10 - *Guidelines on ISO 26262*
- The parts 11 - *Guidelines on application of ISO 26262 to semiconductors* and 12 - *Adaptation of ISO 26262 for motorcycles* followed in 2018.

The main difference between the standards IEC 61508 and ISO 26262 is that the ISO 26262 standard follows the concept of Automotive Safety Integrity Levels (ASIL). The ASIL concept is adapted from the IEC 61508 Safety Integrity Level (SIL) concept. Moreover, the ASIL determination is based on the violation of safety goals which provide requirements to achieve an acceptable level of risk rather than just providing a measure of the reliability of safety functions as it is done for the SIL determination in IEC 61508 [IECc, ISOf]. Furthermore, the IEC 61508 works prescriptively and focuses on safety functions. The ISO 26262 works goal-oriented and focuses on safety goals [DHS15, CDDS10]. Figure 2.5 shows an overview of the ISO 26262 standard and provides evidence of the high complexity of managing the development process and safety life cycle from concept to solution. The V-model is omnipresent in its various parts. Additionally, process steps and deliverables (such as safety plans & goals, bidirectional traceability, and safety case assurance) in combination with the growing complexity of the environment and solutions require new approaches for functional safety management such as model-driven systems engineering. Typical problems in document-centric approaches, such as total traceability, ASIL decomposition and ensuring the consistency of development data e.g. between failure mode and effects analysis (FMEA) and system architecture, can be solved using model-driven engineering [Alt14a]. In contrast to the IEC 61508 standard, model-driven engineering is deeply rooted in ISO 26262. Annex B describes the concept of model-based approaches of in-vehicle software and outlines its implications on product development at the software level. The seamless utilization of models facilitates a highly consistent and efficient development. The standard additionally quotes that these models can be used for simulation or code generation [ISOn, Erk15].

In this context, it should be noted that ISO 26262 addresses functional safety management in a narrow sense and is usually combined with other standards in automotive systems

engineering processes [Alt14b]. Development standards such as the Software Process Improvement and Capability Determination (SPICE) [HDHM06] and its domain-specific derivative Automotive SPICE (ASPICE) [HSDZ+09] define the reference architecture and methodology for the development of (automotive) software systems. Furthermore, the ISO 26262 and ASPICE overlap each other but, nonetheless, these standards are almost always used together in big companies in a way that they complement each other [OFD+17]. Research conducted by Johannessen et al. [JHÖ11] has shown that the ISO 26262 and ASPICE are not used as two separate alternatives. Development standards in the (automotive) industry are not part of this thesis but more information can be found in [Alt12]. Furthermore, Smith & Simpson [SS16] and Ross [Ros14] stated that the ISO 26262 functional safety standard is a non-prescriptive guidance. This means that experience and a certain amount of interpretation effort are required by the user.

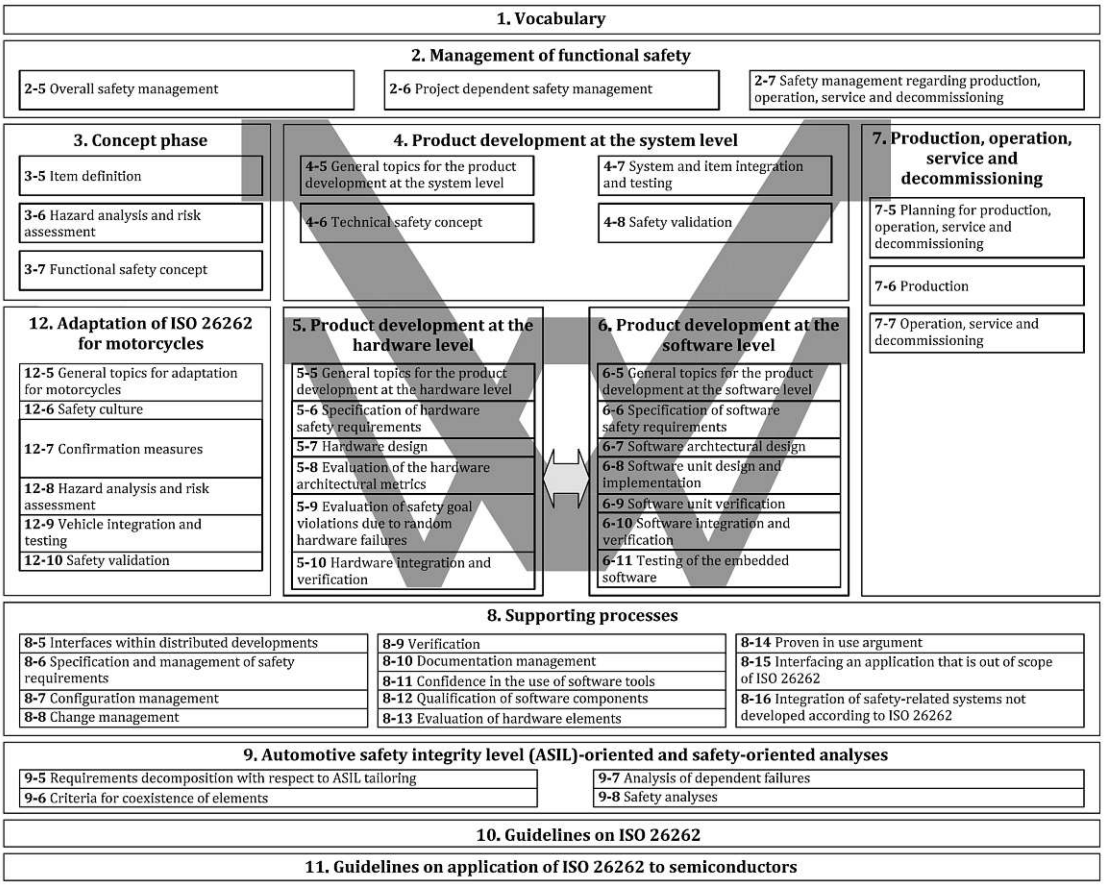

Figure 2.5: Overview of the ISO 26262 entitled *Road vehicles - Functional Safety* [ISOd, p. 7]).

2.2.3 EN 50128 - Software for Railway Control and Protection Systems

The European Committee for Standardization (CEN) is a public association that brings together the national standardization bodies of 39 European countries and was founded in 1961. CEN is one of the three European standardization organizations officially recognized by the European Union (EU) and by the European Free Trade Association (EFTA). The other two officially recognized European standardization organizations are the European Committee for Electrotechnical Standardization (CENELEC) and the European Telecommunications Institute (ETSI). CEN is responsible for the standardization for sectors other than the CENELC and ETSI are designated for - for instance, in matters relating to air and space, defense and security, and transportation. Previous plans for merging the three standardization bodies did not have clear benefits. Thus, the merger has not been done yet [Uni98, Par99, EN:].

The European standard EN 50128 entitled *Software for railway control and protection systems* was first released in 2001 and last updated in 2011. It is a domain-specific variant of the IEC 61508 standard and one of three railway standards for railway signaling. The other two are the EN 50126-1, entitled *The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*, and the EN 50129, entitled *Safety related electronic systems for signalling*. The EN 50126 and the EN 50129 were both last updated in 2018. All three standards are known as standards for *Railway applications - Communications, signalling and processing systems* [EN5, RFT16].

The EN 50128 inherited the concept of the Safety Integrity Level (SIL) from level 0 (lowest) to 4 (highest) and is based on the Tolerable Hazard Rate (THR) per hour and function calculation. Model-based approaches are not explicitly mentioned in the EN 50128 standard. However, formal approaches are recommended for specific Safety Integrity Levels [EN5]:

- SIL 1/2: Formal methods are recommended for software requirement specifications (table A.2), software architecture (table A.3), and modeling (table A.17). Formal proofs are recommended for verification and testing (table A.5). Event tree analysis (ETA) and fault tree analysis (FTA) are recommended for software checking (table A.9).
- SIL 3/4: Formal methods and proofs are identical to SIL 1/2 but they are all highly recommended. Event tree analysis (ETA) is recommended and fault tree analysis (FTA) is highly recommended for software checking (table A.9).

Verification and validation in large and complex railway command and control systems can be accomplished through the introduction of formal methods in the process. Model-driven engineering approaches and formal modeling are promising and efficient methods of showing the correctness of these systems. Figure 2.6 shows how model-driven modeling

capabilities (in green color) map to the artifacts (in yellow color) in the V-cycle. The V-cycle is a V-model iterated for each feature. Furthermore, strongly growing requirements due to system complexity such as documentation and traceability can be addressed by model-based approaches [RFT16].

In summary, this section has shown that model-based approaches are not as grounded in the railway domain as in the automotive domain, but there is a high potential to join formal approaches with model-based approaches to increase safety and efficiency by automation of tasks in systems development processes.

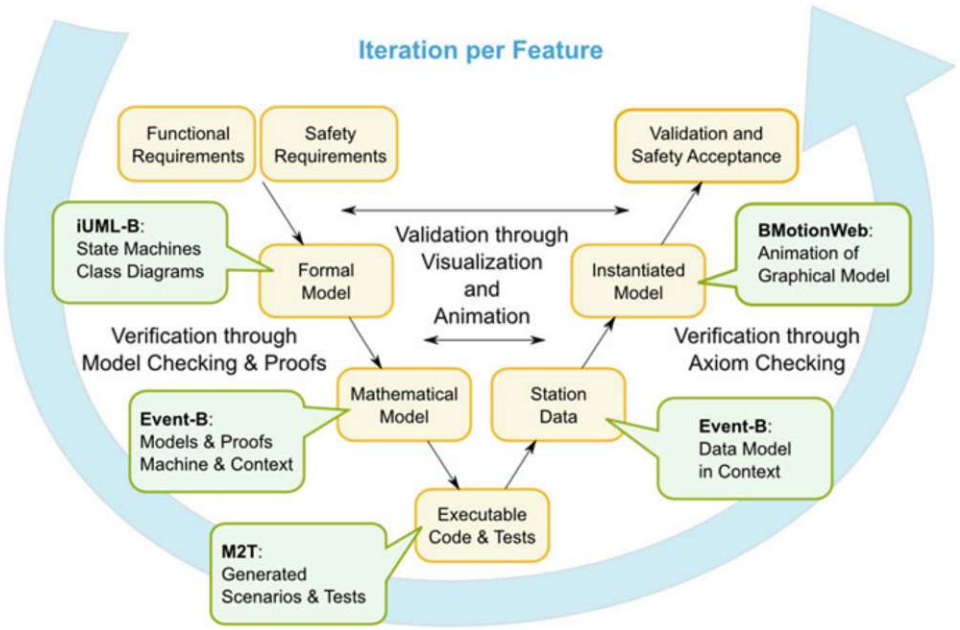

Figure 2.6: Possible artifact mapping to verification & validation cycle within the iterated V-model ([RFT16, p. 4]).

2.2.4 DO-178C - Software Considerations in Airborne Systems and Equipment Certification

In the aerospace industry there are different bodies owning various safety-critical standards, for instance the American Institute of Aeronautics and Astronautics (AIAA), the European Space Agency (ESA), the National Aeronautics and Space Administration (NASA), the Radio Technical Commission for Aeronautics (RTCA), and the Society of Automotive Engineers International (SAE) [ZBL11, SAE]. The de facto standard for the development and approval of commercial software-based systems is the RTCA DO-178, entitled *Software Considerations in Airborne Systems and Equipment Certification* [DO-a, DO-b, HH01]. DO-178 - Version C is the latest release from 2012. Since the

DO-178C standard only focuses on the impact of software failure on the safety of a system, further guidelines are needed which cover the overall development process of a safety-critical aircraft system. The ARP 4754A is an Aerospace Recommended Practice (ARP) from SAE international [SAE] entitled *Guidelines For Development Of Civil Aircraft and Systems* [ARP]. This recommendation serves as a guideline for the development of safety-critical aircraft and systems and addresses the complete aircraft development cycle, from systems requirements to systems verification [Pot12]. Model-based design is a model-based engineering approach and highly recommended in ARP 4754A:

> *"Model use for requirements validation typically uses a model of the environment of a system being developed, which is interfaced to a prototype of a design solution for those requirements. An environment model that is representative of the environment of the system being developed provides a high degree of functional coverage in exercising either a simulated or real system.*
>
> *Models used to capture requirements and then directly used to produce embedded code (Software or HDL*[8]*) come within the scope of DO-178B ... from the time that certification credit is to be taken until the software or hardware is returned to the system processes for system verification"* [ARP].

The ARP 4754A standard is supported by other standards such as the RTCA DO-178C/RTCA DO-178B and DO-254. Figure 2.7 shows the V-model for avionic systems development and how the RTCA DO-178C standard is intended to be used in conjunction with the ARP 4574A for software development.
The IEC 61508 Safety Integrity Level (SIL) concept was adapted and specialized in the DO-178C standard and is called Item Development Level (IDAL). The IDAL comprises four levels: Minor, Major, Hazardous and Catastrophic. Safety assessment processes and hazard analyses examine the effects of a failure condition in the systems and determine the IDAL. The IDAL furthermore sets the number of objectives to be achieved, including "independence" which refers to a separation of responsibilities where the objectivity of the verification and validation processes is ensured by virtue of their "independence" from the software development team. Additionally, DO-178C requires traceability links between the certification artifacts. These links are necessary to ensure that each requirement is fulfilled by source code and successful test cases are linked to the parts of the source code. A traceability analysis ensures that the system is complete as for the requirements determined by the IDAL [DO-b].

The DO-178C is the only standard presented in this thesis which is extended by technology supplements. Furthermore, it is the only one that has a dedicated model-based supplement specifying domain-specific model-based development, rather than addressing the use of model-based development techniques for software development in the main standard as in ISO 26262. An overview of the RTCA document set for the safety of

[8] Hardware Description Language

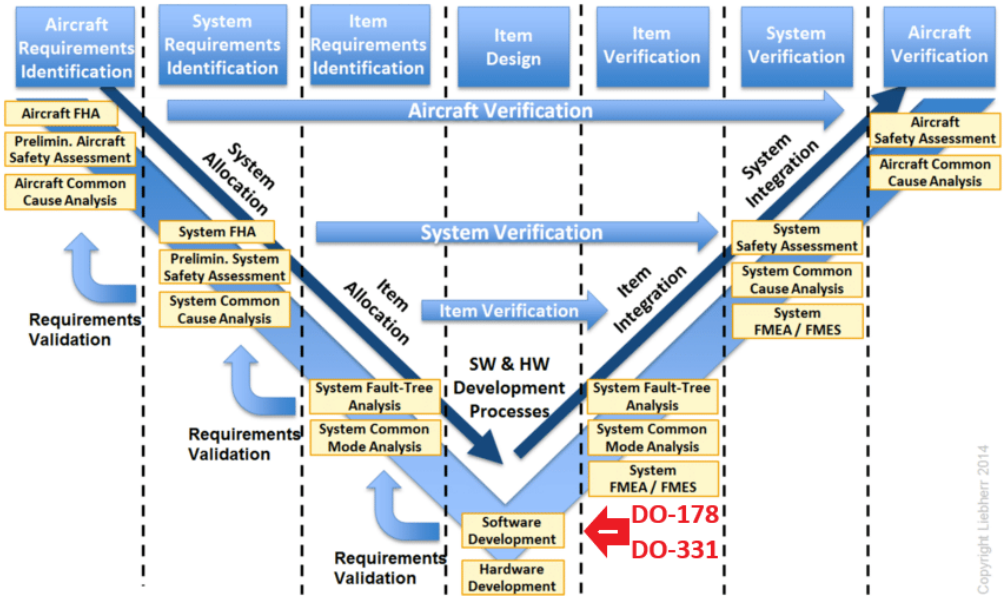

Figure 2.7: ARP 4754A avionic systems development V-model standard (adapted from [TLDP15, p. 127]).

airborne systems is shown in Figure 2.8. The three RTCA guidance supplements for software development are as follows:

- DO-331 *Model-Based Development and Verification Supplement to DO-178C and DO-278A*: This supplement adds and modifies DO-178C for model-based development and model-based design usage [DO-c]. Alford [Afl17] lists five model-based design fundamentals for applying DO-331:

  1. Identify the safe-subset of model-based development technologies to be used in safety-related applications.
  2. Use suitable graphical engineering methods to design a software system.
  3. Make clear distinctions between (graphical) specification and design models.
  4. Define the artifacts which will be in a model, thereby advancing the determination of applicable objectives and activities.
  5. Define model-based development data items to be expected in a program (e.g. model coverage).

- DO-332 *Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A*: This guidance defines further tasks and objectives if object-oriented technology is used as part of the software development life cycle. Furthermore, it provides additional guidance and a framework for evaluating object-oriented technology [DO-d].

- DO-333 *Formal Methods Supplement to DO-178C and DO-278A*: This supplement discusses the use of formal methods for the airworthiness certification of software. Formal methods can contribute to improving the correctness and robustness of the software design [DO-e].

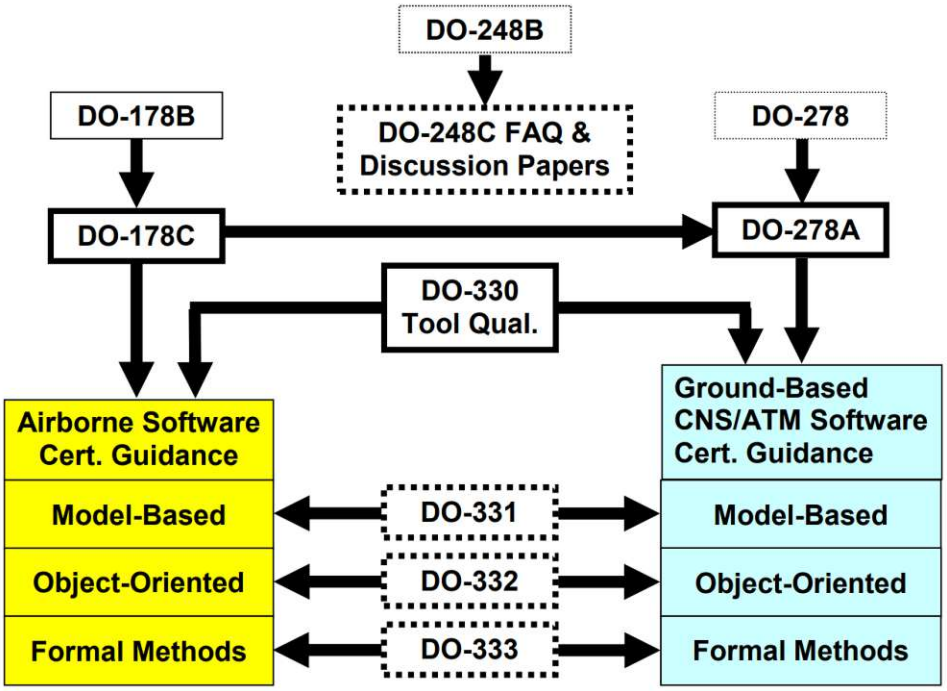

Figure 2.8: RTCA document set for software certification released in December 2011 [Jac12].

As is the case for all presented safety standards in this thesis, assuring DO-178C compliance means high efforts as well as high costs and requires experience to correctly interpret the guidance documents. Further practical information for developing safety-critical DO-178C compliant software can be found in [Rie13].

2.2.5 Summary

This section presented the IEC 61508 standard entitled *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems* and its industry-specific variants ISO 26262 (Automotive), EN 50128 (Railway), and DO-178C (Airborne Systems), in general and in the context of model-based engineering. The IEC 61508 and the EN 50128 do not recommend model-driven engineering approaches explicitly, but research has shown that obligatory formal approaches could be improved by applying model-based engineering methods [RFT16]. In contrast, model-driven engineering is deeply rooted in ISO 26262 and most advanced in the DO-178C standard. The DO-178C has its

supplement DO-331 for safety-compliant model-based development and model-based design usage.

However, all standards share one similarity, namely the V-model. It is ubiquitously comprised to a greater or lesser extent in all standards, and requirements such as traceability and automated verification are challenging tasks in the safety life cycle. If these characteristics will be joined with the ever-growing complexity of safety-critical systems, model-driven systems engineering can be a good solution for taking software development to the next necessary level, regardless of the industry domain.

CHAPTER 3

Systematic Mapping Study

This part of the thesis introduces the two empirical and systematic research methods, Systematic Mapping Study (SMS) and Systematic Literature Review (SLR). The chapter starts with an introduction on the history and state of the art of research in software engineering. Afterwards, information about the SMS is provided and its differences to the SLR are explained in detail. Furthermore, the SMS methodology is explained in more detail to provide background knowledge for our systematic mapping process. In particular, the SMS conducted for the purpose of this thesis focuses on three research questions. The first one aims to identify which safety-critical standards based on IEC 61508 apply model-based methods to what extent (e.g. ISO 26262). The second research question deals with the reasons for applying model-based methods (e.g. traceability). And the third and last one aims at categorizing which model-based methods are applied for the top three most common safety standards from the first research question (e.g. model transformations). Each subsection describes one process step and its outcome in the conducted SMS, concluded by graphical presentations and a detailed discussion about the proposed results. Furthermore, the most interesting publications for each research question and each standard are discussed after the systematic mapping process is completed.

3.1 Introduction and Background

In the early 1950s, the first programming languages started to appear for both research and prototype machines as well as commercially manufactured machines [Cam82]. Major programming languages that are occasionally still used for the maintenance of legacy systems such as FORTRAN or COBOL were released in the late 1950s. The field of application of the languages was to deal with more sophisticated problems in terms of writing code in high-level languages rather than assembly language. It is not clear when the term *software engineering* was first used in scientific research. In the ACM's issue of Commu-

nications from 1966, the term was probably formally used in a scientific publication for the first time [ACM66, p. 546]. Furthermore, the Study Group on Computer Science of the NATO Science Committee called for an international conference on the subject of *software engineering* in 1967 [Mah90]. Aspray et al. [AKP99] stated in 1999 that *software engineering* has not yet achieved the status of a scientific discipline. Furthermore, Glass et al. [GVR02] declared that *software engineering* research has been criticized from several points of view such as that its research is immature and its approaches are unscientific. However, today the research area of *software engineering* matures more and more and specific methodologies have been developed. These methodologies have increased the quality of *software engineering* research significantly. In the area of secondary study methods, the two most popular ones (cf. [PVK15, p. 15]) are the Systematic Literature Review [BC07] and the Systematic Mapping Study [PFMM08]. Although the thesis follows the SMS research guidelines by Petersen et al. [PFMM08, PVK15], a short introduction on both SLR and SMS is given.

The Systematic Mapping Study in software engineering is a defined method to build a classification scheme and structure a software engineering field of interest. This thesis follows the guidelines for conducting an SMS by Petersen et al. [PFMM08, PVK15]. The proposed systematic mapping process (see Figure 3.1) categorizes published research reports and results following predefined activities. The final result is a systematic map, which can be presented using visualization techniques to provide a visual summary of the research question results. Since the release of the first guidelines, the interest in this methodology has grown steadily and the number of mapping studies has continuously increased. The guideline was last updated in 2015 [PVK15]. A detailed discussion about Systematic Mapping Studies and the study conducted in this thesis can be found in section 3.2.

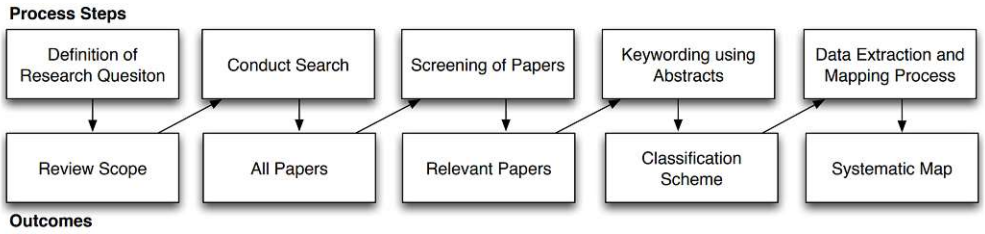

Figure 3.1: The five steps of a systematic mapping process and its outcomes [PFMM08, p. 2].

In the area of model-based engineering, various related works follow the SMS approach for structuring their empirical research. An SMS by Hojaji et al. [HMZ+19] surveyed and classified existing scientific publications about the model-driven engineering approach, the so-called model execution tracing. Beside interesting research results, the study showed that most existing approaches were not validated empirically. Missing validation can raise doubts about the effectiveness of approaches applied in practice.

Domain-specific modeling languages (DSL) were researched and classified in an SMS in the Industry 4.0 domain, which integrates Cyber-Physical Systems with the Internet of Things (IoT) to optimize the complete value-added chain [WBCW19]. The results showed that domain-specific languages from systems engineering such as AutomationML[1] and knowledge representations are applied most often but rarely combined. Furthermore, the authors identified a gap between research communities.

Wolny et al. [WMC+19] conducted an SMS about OMG's Systems Modeling Language (SysML), which has been on the market for more than 13 years. The results revealed that there is a growing interest in the SysML language. Furthermore, a solid basis is provided for classifying existing approaches for SysML. The authors also fully utilized the possibilities and advantages of an SMS to analyze and classify a very high number of scientific publications in a reproducible manner.

Nevertheless, one of the main advantages of an SMS is the possibility to identify research trends or gaps in certain research areas. An SMS about gamification in education showed that classification is also possible for combining different research areas, more precisely, a combination of design principles, gamification, and educational purposes. The results revealed that extrinsic and intrinsic motivation of the learners can be influenced by gamification [DDAA15].

A more extensive guideline for undertaking systematic reviews in the software engineering field of interest is the Systematic Literature Review invented by Kitchenham & Charters [BC07]. An SLR is usually conducted by a research team and not by a single person due to its extensive workload compared to an SMS. The difference between these two approaches is that in an SLR an analysis of the primary studies is done, followed by a data synthesis and a reporting of the results (see Figure 3.2), while in an SMS a plain classification is sufficient. In the model-engineering area of research a few Systematic Literature Reviews have been conducted, some of which we will discuss below.

Carroll & Malins [CM16] asked the question of how the change from a document-based systems engineering approach to a model-based systems engineering approach is justified in the defense, space, and complex systems product engineering industries. The SLR covered 67 case studies without metrics and 21 case studies with metrics on cost and schedule. The primary conclusion was that the shift to a model-based systems engineering approach resulted in a significant advantage for project performance.

An SLR by Sales & Becker [SB18] researched the systems engineering design methods, standards, process models, and their employment in model-based systems engineering. The results showed that there are difficulties in using only model-based engineering approaches because most complex systems are dependent on various standards and

[1] Automation Markup Language [KWS+15].

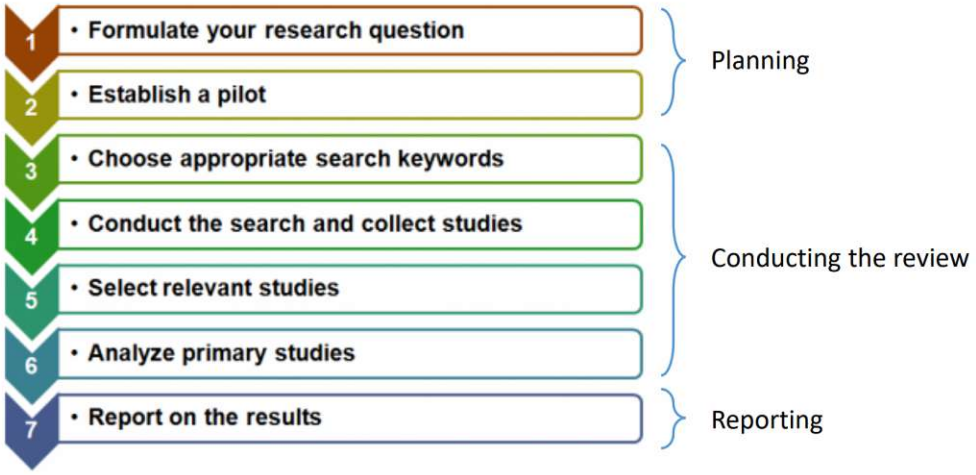

Figure 3.2: Systematic Literature Review (SLR) process [Sch19].

their key aspects. Another SLR in the domain of requirements engineering & modeling investigated which modeling methods have been studied and how well these studies have been conveyed [YLJC14].

Systematic Literature Reviews cannot only be used for research on model-based research questions. It is also possible to use model-based approaches to tackle the disadvantages of SLRs. Barat et al. [BCBK17] addressed the issues effort, time, and intensive intellectual endeavor of SLRs, and they proposed a model-based approach for conducting systematic reviews of research literature. The created domain-specific language led to a reduction in time, effort, and the current dependence on human expertise. The evaluation was based on a real-life case study evaluation. Current research is exploring the use of natural language processing (NLP) to further improve the automation of SLRs.

In summary, an SMS is used to structure a research area, while an SLR is focused on gathering and synthesizing evidence.

3.2 Activities

In this section, the five steps of the systematic mapping process based on the research topic of our survey are explained and discussed in detail. The process for conducting this SMS is based on the guidelines published by Petersen et al. [PFMM08, PVK15]. The five process steps are shown in Figure 3.3.

The first step of the systematic mapping process is the definition of research questions. The output of this process are research questions that define the review scope for the second step of the process. In this second step, the literature search is done, and certain

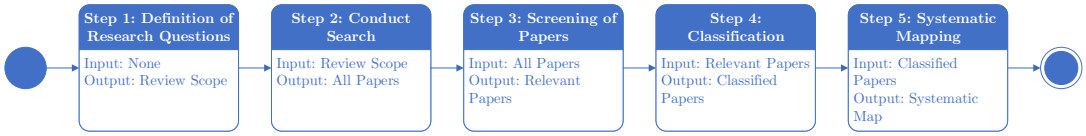

Figure 3.3: Overview of the SMS process steps.

decisions such as the selection of relevant libraries for the scientific publications are made. In the third step, selected scientific publications are screened based on filter criteria, and relevant publications are included in the result set for the fourth step. This fourth step, *Classification Using Abstracts*, is based on research facets and enhanced for the purpose of this thesis because in order to answer the research questions more information than the information contained in the abstract is required.[2] The output of the fourth step are classified publications, which are the input for the fifth and last process step, *Systematic Mapping*. In this final process step, a systematic map is created to enable the extraction of information and answer the defined research questions.

3.2.1 Defining Research Questions

In this subsection, the three research questions are defined to specify the review scope. The definition of the research questions is the first activity in this SMS (see Figure 3.4). Furthermore, the intentions behind the three research questions are explained.

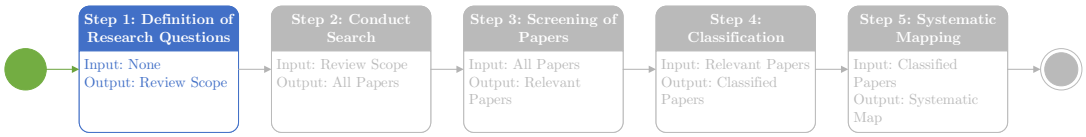

Figure 3.4: Step 1/5: Definition of Research Questions.

- **RQ 1:** *In which safety-critical domains based on/related to IEC 61508 are model-based methods applied? Which standards for functional safety are applied?*
  The first research question intends to get a quantitative result of the model-based engineering applications in different safety-critical domains. The problem statement for this question is that there is no big picture of the degree of utilization of model-based methods in IEC 61508 and standards based or related to IEC 61508. There are a great many scientific publications but research pursues narrow research goals for specific standards rather than research goals across different industry domains or safety standards. Thus, the derivation of results aims to find out if

[2] Enhancements of specific SMS process steps were also done by other authors such as Wolny et al. [WMC+19] to improve the final results.

some safety-critical domains use model-based engineering to a great extent, not at all, or somehow in-between.

- **RQ 2:** *What are the reasons for applying model-based methods in IEC 61508 and each of the top three most common safety-critical standards in RQ 1?*
  In this research question, we are interested in finding out which problems should be tackled or solved by applying model-based methods. For instance, *traceability* is a well-known safety-critical problem in the automotive software development domain and addressed mostly by model-based engineering [Adl14]. The categorization of the methods is done by a self created facet used for IEC 61508 and each of the top three most common model-based standards from RQ 1[3]. Since all model-based domains examined in this SMS are based on the parent standard IEC 61508, we want to know whether the reasons are either mostly the same across all standards or if industry-specific variants imply industry-specific differences in the reasons.

- **RQ 3:** *Which model-based methods are applied in IEC 61508 and each of the top three most common safety-critical standards in RQ 1?*
  The third and last research question intends to categorize the model-based methods applied in different publications by the main paradigms of model-based engineering. Therefore, a designated facet is created to categorize publications by a specific schema. This facet is used for IEC 61508 and each of the top three most common model-based standards from RQ 1[4]. Based on this facet, the similarities and differences of model-based methods between safety-critical standards and domains can be analyzed. Since these similarities and differences affect the tool utilization to a great extent, the results could be useful for tool selection decisions or further research.

3.2.2 Conduct Search

After identifying the research scope and the research questions, the next step is the identification of primary studies using search strings on scientific databases completed with manual browsing (see Figure 3.5).

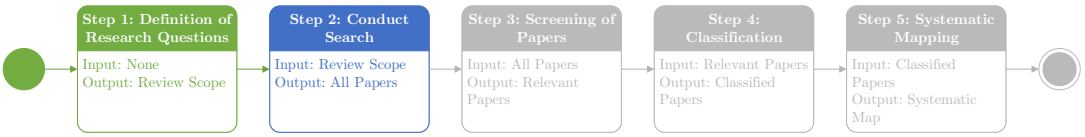

Figure 3.5: Step 2/5: Conduct Search.

Since there are many digital libraries for scientific publications, the selection is based on an expert interview with a professional experienced computer science and Systematic

[3] In this case, the top three most common model-based standards are the three most occurring results from RQ 1.

[4] See footnote above.

Mapping Study researcher. A coarse search was done for the four functional safety standards, namely IEC 61508, ISO 26262, EN 50128, and DO-178(B/C). The following sources were selected for publications:

1. Scientific Databases:

   - IEEE[5]: The IEEE Xplore digital library provides access to scientific and technical content published by the IEEE (Institute of Electrical and Electronics Engineers). The digital library contains books, conferences, courses, journals and magazines, and standards. All IEEE conference papers go through a peer-review process before publication.

     Search Query (case-insensitive, ignoring "-"):
     ("Placeholder for functional safety standard") AND ("model based software engineering" OR "model based systems engineering" OR "model driven software engineering" OR "model driven systems engineering" OR "Modellgetriebene Softwareentwicklung" OR "Modellgetriebene Systementwicklung" OR "Model Driven Architecture" OR "Model Driven Engineering" OR "Modellbasierte Entwicklung")

   - Scopus[6]: Scopus is a source-neutral abstract and citation database specialized in scientific, technical and medical content provided by Elsevier [Els]. All journals covered in the Scopus database are peer-reviewed to ensure top-level quality standards.

     Search Query (case-insensitive, ignoring "-"):
     ("Placeholder for functional safety standard") AND "model based software engineering" OR "model based systems engineering" OR "model driven software engineering" OR "model driven systems engineering" OR "Modellgetriebene Softwareentwicklung" OR "Modellgetriebene Systementwicklung" OR "Model Driven Architecture" OR "Model Driven Engineering" OR "Modellbasierte Entwicklung"

   - Springer Link[7]: Springer Link is an interdisciplinary portal that offers a variety of information resources on all subject areas such as computer science and engineering. The database is hosted by Springer Nature, an academic publishing company and provides access journals, books & book series, protocols, and proceedings.

[5] http://ieeexplore.ieee.org/
[6] http://www.scopus.com/
[7] https://link.springer.com/

Search Query (case-insensitive, ignoring "-"):
("Placeholder for functional safety standard") "model based software engineering" OR "model based systems engineering" OR "model driven software engineering" OR "model driven systems engineering" OR "Modellgetriebene Softwareentwicklung" OR "Modellgetriebene Systementwicklung" OR "Model Driven Architecture" OR "Model Driven Engineering" OR "Modellbasierte Entwicklung"

- DBLP[8]: The database and logic programming bibliography site is a computer science bibliography. DBLP lists journal articles, conference papers, and other publications on computer science.

  Search Query (case-insensitive, ignoring "-"):
  (Placeholder for functional safety standard) Model

- DOMA[9]: A literature database with bibliography, abstract, and keywords for mechanical engineering, plant engineering, and manufacturing technology. The DOMA database is part of the TEMA® database and hosted by the WTI Frankfurt organization [WTI].

  Search Query (case-insensitive, ignoring "-"):
  (Placeholder for functional safety standard) Model

2. Google Scholar: Because Google Scholar does not support long search queries, a coarse search for each of the four standards was done on the first three result pages. The goal was to find papers that are not published on the selected scientific databases but are of great importance to the SMS. For instance, the experience paper from Born et al. [BFK10] was found on Google Scholar.

3. Domain expert: A few publications about ISO 26262 were recommended by an expert in the automotive software development domain. All publications were included in the third and next step, which is called screening of papers.

4. References: During the screening process interesting references for our field of interest were found in the screened papers. If the referenced publications were available and accessible, they were included in the pool of all papers and also screened.

For the sources Google Scholar, domain expert, and references, the authors of the publications were cross-checked to ensure their scientific background. Since these three

[8] http://dblp.uni-trier.de/
[9] http://http://wtiweb.wti-frankfurt.de/

	Safety Standards for Functional Safety			
Databases	IEC 61508	ISO 26262	DO-178(B/C)	EN 50128
DBLP	9	12	4	0
IEEE	2	5	5	0
Scopus	128	161	153	21
Springer	86	114	94	29
Miscellaneous	5	15	5	3
Total	**230**	**307**	**261**	**53**
Access Date	23.10.2019	26.10.2019	02.01.2020	02.01.2020

Table 3.1: Total number of publications after conduct search.

literature sources are only a small part of our total result set, they are addressed as *Miscellaneous* in the next parts of the thesis. One objective of this research is to provide a comprehensive overview of standards and model-based engineering methods rather than just cover a single aspect of functional safety engineering. Thus, the search keywords cover as many possibilities as deemed useful. Furthermore, no restrictions on the release periods were included. The total number of the final result set is listed in Table 3.1.

3.2.3 Screening of Papers

In the third process step, all publications are screened and either excluded or included in the pool of relevant papers. The pool of relevant papers serves as input for the classification process step (see Figure 3.6). For proper screening, exclusion criteria were used to exclude publications that are not relevant to answer at least one of the research questions.

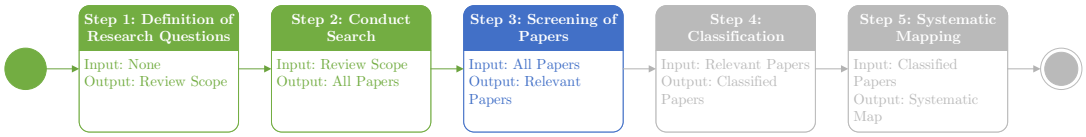

Figure 3.6: Step 3/5: Screening of Papers.

The defined exclusion criteria are as follows:

- Duplicates.
- Text in other languages than German or English.
- No specific context to the defined safety standards for functional safety. The publication must outline that the defined safety standard is addressed rather than just introduced.

- No specific context to the model-based engineering or model-driven engineering domain. For instance, model-based approaches are only addressed as related work but are not the main focus of the publication.
- For papers only:
  - Abstract and/or conclusion are out of scope.
  - Identical abstracts. Some papers are published with identical content at different venues with different release dates such as conferences and journals. In this case, all but one paper are excluded.

Several scientific studies also excluded non-peer reviewed publications [PVK15]. Since an important amount of publications in the pool are books, non-peer review publications are not excluded in this SMS. Based on the defined exclusion criteria, all publications were screened and relevant ones were selected. The number of relevant papers is listed in Table 3.2. The overall list of all relevant publications, separated by standard and source, is provided in Appendix 5.

	Safety Standards for Functional Safety			
Process Step	IEC 61508	ISO 26262	DO-178(B/C)	EN 50128
All Papers	230	307	261	53
Excluded	126	160	184	30
Total	**104**	**147**	**77**	**23**

Table 3.2: Total number of relevant publications after screening of all papers.

3.2.4 Classification

In this process step, the development of the classification scheme and the execution of the classification process based on keywording is described (see Figure 3.7). The guidelines for conducting an SMS by Petersen et al. [PFMM08] described a systematic keywording process based on the abstracts to build a classification scheme. Since the relevant publications also included books, it was necessary to adapt to the proposed process. The keywording process for books was applied to relevant chapters identified during the conduct search process step. Furthermore, sometimes the use of abstracts and conclusions during the classifications of papers was not sufficient. In this case, specific chapters of the paper were identified and also included in the keywording and classification process.
In this SMS, three final sets of keywords were chosen. These three sets of keywords allow the creation of categories for the systematic map. In order to answer the three research questions, the mapping process is based on self created categorizations as described below:

Safety-critical domain background: The definition of the categories was encouraged by the industry sectors based on the safety standard IEC 61508 and discussed by

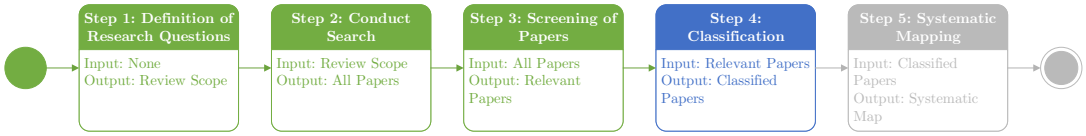

Figure 3.7: Step 4/5: Classification.

Smith & Simpson's work [SS16]. Based on the domain background, the following nine categories were defined:

- *Automotive:* Specific electronic systems for road vehicles, tractors, and machinery for agriculture.
- *Airborne Systems:* Airborne electronic systems used in civil avionics and commercial software-based aerospace systems.
- *Factory:* Machinery and robotics in production and factory environments.
- *Railway:* Programmable electronic systems for railway and railway-related applications.
- *Medical:* Medical electrical equipment and electromagnetic compatibility of medical equipment (EMC).
- *Maritime:* Electronic maritime systems for surface vessel and submarine systems.
- *Nuclear:* Electronic instrumentation and control systems in nuclear power plants important to safety.
- *Real-time embedded systems:* General real-time embedded systems such as control units which are not related to one of the specific application contexts above.
- *Miscellaneous:* A categorization is possible but it does not correspond to one of the eight categories mentioned above and is not important enough to create a separate category.

Safety-critical standards for functional safety: Due to the fact that this research is mainly focusing on the functional safety standard IEC 61508 and its industry- and application-specific variants, all other standards are not listed as a separate category. Exceptions are the IEC 61499 and ISO 14971, which are not based on IEC 61508 but occurred in the systematic mapping process. This SMS distinguishes between the following safety standards:

- *IEC 61499:* Function Blocks [IECa]

- *IEC 61508:* Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems [IECb].
- *ISO 14971:* Medical devices — Application of risk management to medical devices [ISOa].
- *ISO 26262:* Road vehicles – Functional safety [ISOd].
- *DO-178(B/C):* Software Considerations in Airborne Systems and Equipment Certification [DO-a, DO-b].
- *EN 50128:* Railway applications - Communications, signaling and processing systems - Software for railway control and protection systems [EN5].
- *Miscellaneous:* A categorization is possible but it does not correspond to one of the four categories above and is not important enough to create a separate category.

Reasons for applying model-based engineering methods: Since the system and safety requirements differ from safety standard to safety standard, a clear difference in the methods applied per security standard is also expected. This category covers all minor and major reasons for applying model-based engineering methods for the safety standards IEC 61508, ISO 26262, DO-178(B/C), and EN 50128. The twelve reasons are listed and described below:

- *Change Management:* Models are used to enable and support consistent change management.
- *Communication:* Improvement of communication between safety stakeholders in the conceptualization or development phase. Safety stakeholders are safety engineers, software engineers, and certification authorities [ZBL11].
- *Collaboration:* Better team collaboration and cross-organizational data exchange. This category also includes publications on problem solutions of collaborative modeling such as data management and synchronous modeling.
- *Consistency:* Model-based engineering methods that are applied for improving consistency or avoiding inconsistency. This category furthermore includes the enhancement of model checking, namely consistency checking, to detect inconsistencies between models.
- *Quality Management:* A more general reason for applying model-based engineering methods is quality management. This category implicitly includes several other reasons outlined in this list, but the authors mentioned the more generalized reason quality management rather than detailed reasons.

- *Requirements Engineering:* The definition, managing, and traceability of defined (safety) requirements is a complex phenomenon when working with safety standards. Publications addressing a problem regarding requirements with model-based engineering methods are included in this category.

- *Re-use:* The re-use of proven solutions for safety-critical applications is considered good practice for increasing confidence in the system design and cutting development cost and time and is widely spread in practice [Kha18]. Publications addressing modularity and re-use of solutions with model-based engineering methods are assigned to this category.

- *Safety Analysis/Certification:* All publications addressing the issue of ensuring safety and safety assessment based on safety requirements are listed in this category. Furthermore, publications focusing on the fulfillment of safety compliance needs for the underlying functional safety standard are also listed in this category.

- *System Development:* In this category, model-based engineering methods are applied for a big part or all parts of the overall system development. Model-based system development is mostly combined with other reasons such as requirements engineering or safety analysis/certification.

- *Testing:* This category includes the purpose of using model-based engineering methods for testing, also known as model-based testing. The testing workflow of the model-based development of safety-related software is mostly divided into two steps, requirements-based testing and back-to-back testing as shown in Figure 3.8 [Con09, Bei10, Con12]. However, this category includes all publications irrespective of whether they pursue common or individual model-based testing approaches.

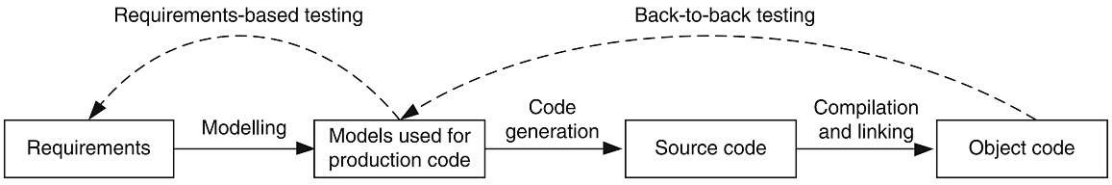

Figure 3.8: A general workflow for the model-based development of safety-related software [FAS+15].

- *Traceability:* All safety standards investigated, namely the IEC 61508, ISO 26262, DO-178(B/C), and EN 50128, recommend a V-model for system development. All V-models include links for validation, verification, or satisfaction from higher to lower levels and vice versa or links across levels from left to right and vice versa. This category includes all publications which use or intend to use model-based methods to ensure or analyze traceability.

- *Workflow Management:* When working with safety-critical standards the first step is to create an artifact called a safety case [Kel04]. Further steps can be defined in a workflow based on a model. Workflow models can furthermore be included in a megamodel [SKDS+19]. All proposed workflows regardless of their specific implementation supported by model-based methods are assigned to this category.

Applied model-based engineering methods: The possibilities to apply model-based engineering are manifold. The reasons for applying model-based engineering methods are strongly associated with the applied methods themselves. The reasons can be seen as a problem definition and the applied methods can be seen as a possible problem solution. The eight categories cover almost all methods applied in the relevant publications and are as follows:

- *Code Generation:* Model-to-Text transformations are used for automating tasks such as documentation, reporting, task lists, etc. This category comprises M2T transformations with code generation only. Thus, the transformation is a transition from the model level to the code level and requires two different meta-models.
- *Domain-Specific Modeling Language (DSL):* The development or enhancement of a DSL is explicitly mentioned in the publication. Meta-model modeling and UML profiles can be considered as DSL too. However, the definition of a DSL is complex and several principles must be followed in order to define a useful and good domain-specific modeling language [BCW17, SZ09].
- *Meta-Model Modeling:* A specific meta-model is defined or an existing one is taken or enhanced. This category can be distinguished from DSL due to the fact that the publications address narrow research goals rather than elaborate principles of domain-specific languages.
- *Model Transformations:* A transformation is performed between a source and a target model on the model level. This category does include both Model-to-Model (M2M) or Model-to-Text (M2T) transformations.
- *Model Validation:* A model is validated using constraints or formal methods to allow the identification of a safe or not-safe model state, a safety standard compliance, or other.
- *Plain Modeling:* An unmodified modeling language without any specific application context is used. For instance, the general-purpose Unified Modeling Language (UML) supports plain modeling and can be used for modeling purposes of any sector or domain.
- *Simulation:* Simulation models are synthesized and executed to evaluate and test a system. Since simulation is typically done in separate domain-specific tools, a more detailed distinction could be done in future research work [BRB17].

- *UML Profiles:* UML profiles are used as an extension mechanism for customizing UML models or as a domain-specific interpretation of UML for refining standard semantics. This category includes both standardized UML profiles, such as Systems Modeling Language (SysML) or Business Process Model and Notation (BPMN), and specific profiles designed and applied by the authors of the publications [OMGb].

The outcome of this process step called *Classification* are classified papers based on the four categories defined above. In summary, four categories and several subcategories were defined. The first category *Safety-critical domain background* and the second category *Safety-critical standards for functional safety* only focus on the first research question. After the first research question has been answered and categorization has been completed for these first two categories, the three most common domains and their underlying safety standards based on IEC 61508 are selected. The IEC 61508 and the three most common application-specific standards based on IEC 61508 are then categorized based on the third category *Reasons for applying model-based engineering methods* and the fourth category *Applied model-based engineering methods*.
In the next section, the categorized publications serve as an input during the process step called *Systematic Mapping*.

3.2.5 Systematic Mapping

The objective of the process *Systematic Mapping* is a systematic map to illustrate the occurrences of publications for each category based on the classified papers (see Figure 3.9). The results of the systematic map have two major goals. Firstly, to see which categories have been emphasized in past research. Secondly, based on categories that are not emphasized, gaps in past and present research can identify possibilities for future research. Furthermore, the analysis of the results to answer the three research questions is described. Each answer for each research question is discussed separately in the following subsections. Additionally, the most interesting publications of the most common categories are introduced in order to provide a good overview of the analyzed publications.

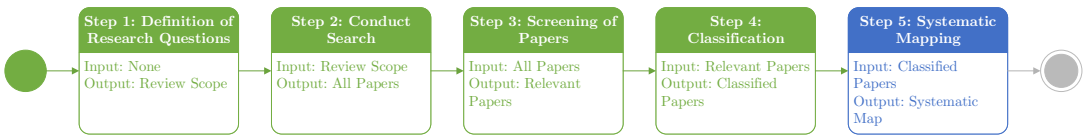

Figure 3.9: Step 5/5: Systematic Mapping.

3.2.5.1 RQ 1: In which safety-critical domains based on/related to IEC 61508 are model-based methods applied? Which standards for functional safety are applied?

In order the answer the first research question, the industry background and safety standard of all 104 publications selected with the IEC 61508 search query were analyzed and the publications were filtered based on the defined criteria. Since the IEC 61508 safety standard has more than 12 industry- and application- specific variants, it was important to find out which variants are used in which domain combined with model-based engineering approaches and to what extent [SS16].

Figure 3.10 depicts the absolute number of occurrences of the domain background in the relevant publications. It is important to note that some publications address several industry domains at once, thus multiple categorizations are possible. The result set shows that research focuses on the *Automotive* domain. *Automotive* is as popular as the five most common other categories, namely *Avionic, Factory, Maritime, Medical, and Railway*. Furthermore, more publications address an industry-specific domain rather than a general-purpose one. The other research questions are based on the IEC 61508 and the three most common IEC 61508 application-specific variants. Thus, this analysis provides the necessary starting point for the second and third research questions. Aside from that, the results about the domain background give great insight into the currently pursued research goals of model-based engineering in the area of functional safety. In addition, it was the intention to find out if the domain backgrounds also match their related specific safety standard variants for functional safety.

Figure 3.11 illustrates the standards applied for functional safety in the pool of 104 relevant publications. It is important to note that some publications address several safety standards at once, thus multiple categorizations are possible in the same way as in Figure 3.10.

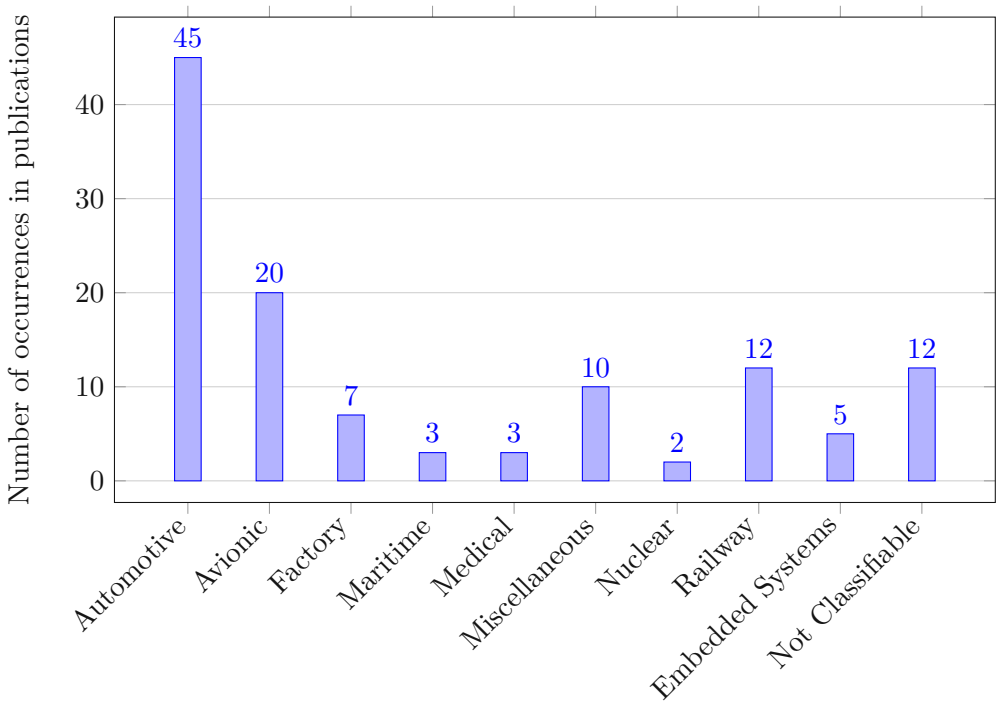

Figure 3.10: RQ1: IEC 61508 safety-critical domains.

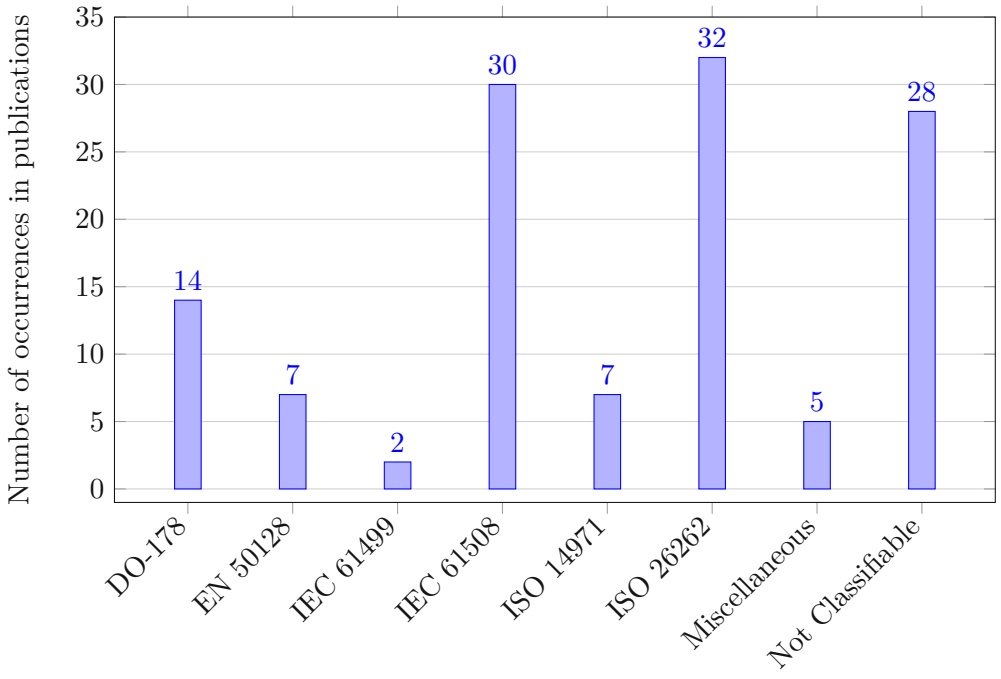

Figure 3.11: RQ1: IEC 61508 and safety standards related to/based on IEC 61508.

In this context, the research question focuses on the crucial question if the domain matches its related safety standards. For instance, it would make sense if the ISO 26262 standard entitled *Road vehicles - Functional safety* is applied in the *Automotive* domain rather than the generic IEC 61508 standard. Table 3.3 outlines the relationship between the occurrences of the industry-specific domain and the related safety standard. The table shows that only a part of the publications apply the industry-specific safety standard related to the domain. Furthermore, the ISO 26262 standard is applied most and matches the most occurring *Automotive* domain. Another interesting fact is that 28 publications do not even address the applied standard in their work.

Relation	**Domain (Figure 3.10)**	**Safety Standard (Figure 3.11)**
Automotive - ISO 26262	45	32
Avionic - DO-178	20	14
Factory/Embedded Sys. - IEC 61499	12	2
Medical - ISO 14971	3	7
Railway - EN 50128	12	7
Miscellaneous - IEC 61508	10	5
Not-Classifiable	12	28

Table 3.3: Relations between domain backgrounds and applied safety standards. The results are based on the IEC 61508 result set.

In summary, the three most common domains in the result set are *Automotive, Avionic and Railway*. In addition, the safety standards applied in the publications were analyzed to answer the question if the generic IEC 61508 standard is also used for industry-specific research. In most of the cases in the result set, industry-specific variants of the IEC 61508 safety standard are applied rather than its generic parent. The most common ones are ISO 26262, DO-178, and EN 50128. These three are further investigated in RQ 2 & RQ3.

3.2.5.2 RQ 2: What are the reasons for applying model-based methods in IEC 61508 and each of the three most common safety-critical standards in RQ 1?

For the categorization process of this research question the definition of categories as explained in the section 3.2.4 was used. In the categorization process of RQ 1, all publications were categorized according to the industry domain and to the safety standard. The three most common safety standards are *ISO 26262, DO-178, and EN 50128*. Thus, search queries were used for the second research question including publications addressing the standards ISO 26262 (Automotive), DO-178 (Avionic), and EN 50128 (Railway), besides the generic IEC 61508. In total, 351 publications of four safety stan-

dards were categorized in order to provide reasoning for applying model-based methods across different safety standards. During the entire process multiple categorizations are possible for each publication.

IEC 61508: We started to deal with the IEC 61508 standard and included 104 relevant publications in the categorization process. Figure 3.12 illustrates the number of occurrences of stated reasons in publications addressing the IEC 61508 standard. Each bar represents one of the eleven reasons. The reasons *Miscellaneous* and *Quality Management* did not occur in the IEC 61508 publications.

Safety Analysis/Certification resulted to be the predominant reason compared to the other ten reasons. Almost every second publication mentions this reason explicitly or implicitly. In [AMYL15], the authors proposed a methodology and associated framework for multi-paradigm model-driven security analysis. In this approach, attack trees are automatically generated based on the results of their proposed security analysis method. The framework is implemented into the Papyrus environment based on UML profiles. Vara & Panesar-Walawege [dlVPW13] presented a multi-domain meta-model called SafetyMet for safety standards targeting the facilitation of safety compliance. Since safety compliance is a highly expensive and demanding task, their meta-model can help to show the fulfillment of the safety criteria specified in the safety standards. The SafetyMet meta-model has been validated with IEC 61508, ISO 26262, EN 50128, and DO-178C.

System Development is the second most common reason for using model-based methods in the IEC 61508 publications. 24 out of 104 publications are using model-based methods for all or most of the *System Development* process. The justification for this reason is complex, since the system development process can be executed or supported in various ways. The paper proposed by Mayr & Saft [MPS11] shows the need for an operational quality model for applying the safety standard and for giving guidance in the system development process for safety-critical software. Furthermore, a specially created quality model for the development process is introduced.

Since *Traceability* is a key requirement of the IEC 61508 standard, it is an unexpected result that only less than a fifth of the publications address this requirement by means of model-based methods. For all other categories it is not possible to identify any patterns, since they are only mentioned occasionally.

ISO 26262: The second standard which was investigated in order to discover the reasons for applying model-based methods is the ISO 26262 safety standard. In the categorization process, 147 relevant papers were included. The reason *Safety Analysis/Certification* is mentioned almost as many times as in the IEC 61508 categorization results. Besides the reason *Safety Analysis/Certification*, the results are not as clear as for the IEC 61508 standard because six different reasons occurred 20 or more times in the systematic mapping process.

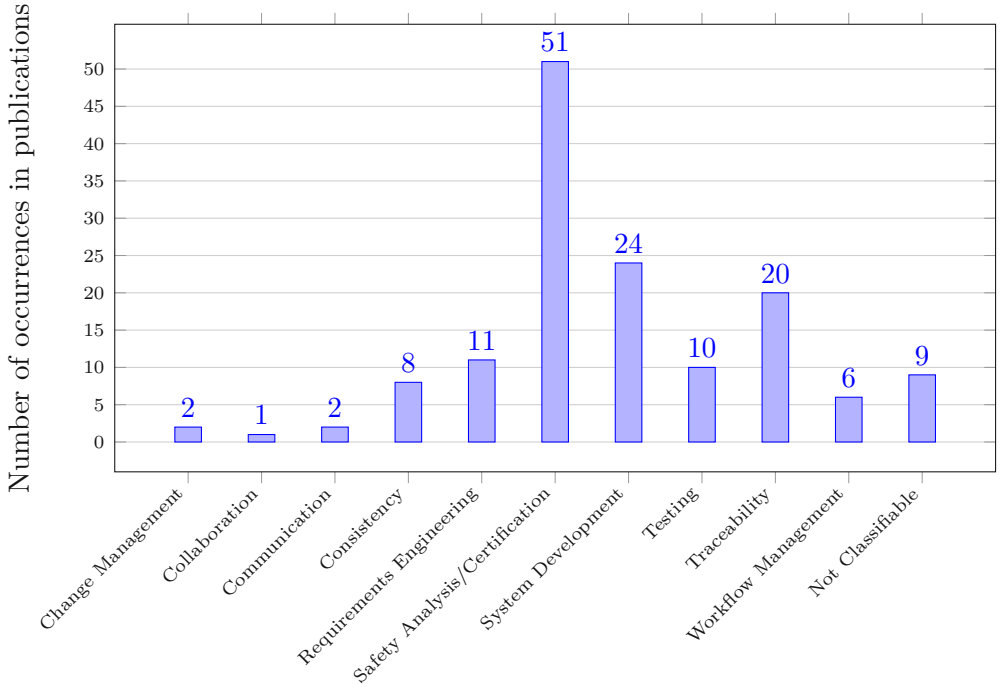

Figure 3.12: RQ2: Reasons for applying model-based methods in the context of IEC 61508 (Generic).

The category *Safety Analysis/Certification* includes various activities. Beckers et al. [BCF+15] presented a model-based method with a UML profile to plan the required safety validation and verification process. The OCL checks developed in the work can furthermore ensure consistency in the validation and verification activities. The authors of [MAL+11] presented an ISO 26262 conform modeling framework. The framework uses the Electronics Architecture and Software Technology - Architecture Description Language (EAST-ADL), a domain-specific language for annotating models and automated detection of errors. Furthermore, a fault tree analysis (FTA) and a failure modes and effects analysis (FMEA), which are both recommended in the ISO 26262 standard, can be automatically synthesized from the same model.

A key requirement of the ISO 26262 is (bidirectional) traceability (see section 2.2.2). The reason *Traceability* is the second most frequently stated reason in the categorized result set. The manual creation of trace links is an error-prone task. Sporer et al. [SMHK15] introduced a model-based development process to ensure traceability from customer requirements until the final release of the product. The developed fully automatic transformations from system design models into software framework models and vice versa guarantee full and bidirectional traceability between system and software level.

Heisig et al. [HSBS19] stated that existing traceability solutions are either limited to specific development processes, tools or do not consider variability. The paper discusses the establishment of uniform traceability-enabled workflows in a variability-aware and model-based environment. Thus, traceability is still being researched extensively and lacks perfect solutions.

A new and relatively often mentioned reason is *Re-Use* in the road vehicles' safety environment. The re-use of artifacts is not just limited to parts of the software. Luo et al. [LBE+13] proposed a model-based approach for assuring compliance with the ISO 26262 safety standard. The authors focused on enabling safety assurance re-use in assessment, qualification and certification processes through objective and cost-efficient modeling of the ISO 26262 standard. Another advantage of the model-based approach is the collection of results and automatically generated report documents. In [Gal14], the author introduced a model-driven safety certification method called MDSafeCer to enable the creation and re-use of process-based arguments. The MDSafeCer objective is to reduce cost and time during the certification process in the context of process models for ISO 26262.

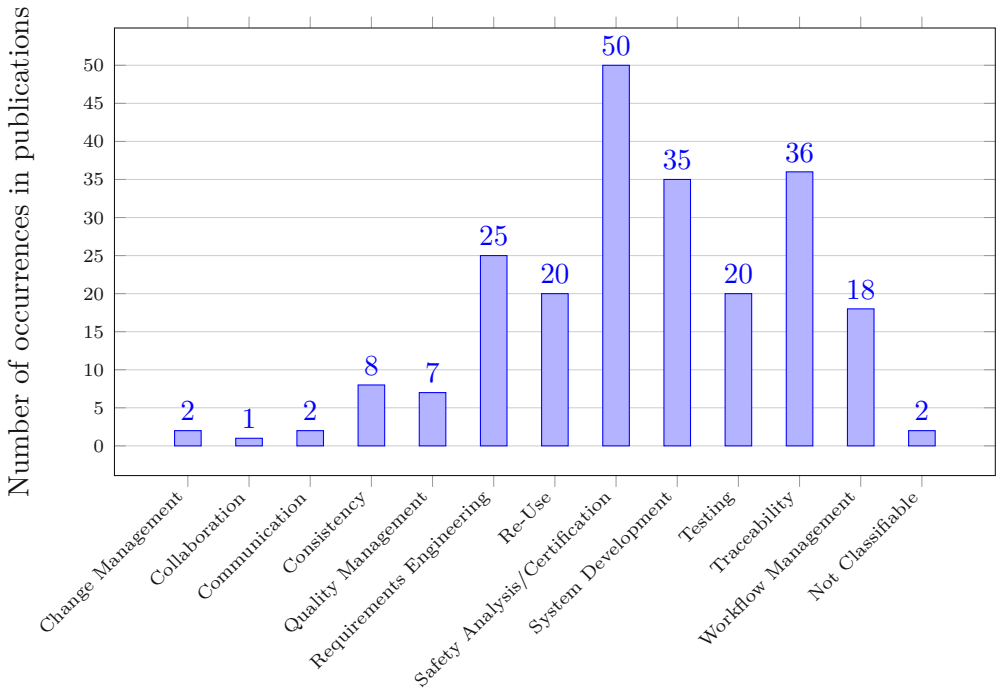

Figure 3.13: RQ2: Reasons for applying model-based methods in the context of ISO 26262 (Automotive).

DO-178: During the systematic mapping process addressing the DO-178 standard, entitled *Software Considerations in Airborne Systems and Equipment Certification*, 77 publications were analyzed and categorized. The results for the different reasons for applying model-based methods in the publications are shown in Figure 3.14. Furthermore, the main findings and the differences to other standards are described below.

As with the other standards, the reason *Safety Analysis/Certification* is the most common one. Furthermore, the reason *System Development* is among the top three in the DO-178 publications and the reason *Testing* became much more interesting in the context of this safety standard.

Multiple publications address the reasons *System Development* and *Safety Analysis/Certification*. Jafer et al. [JDA+18] introduced a model-based approach for developing avionic safety-critical systems throughout the entire software development life cycles. Gaeta & Czarnecki [GC15] described a model-based engineering method and a pattern catalog for modeling avionic systems in SysML. Their evaluation showed that SysML is sufficient for systems design in the aircraft engine domain. The authors of [GD15] presented a model-based development methodology for DO-178C compliance. The model-based development methodology for systems development is based on UML diagrams and model transformations into the formal Z notation to evaluate safety standard compliance.

The biggest and most obvious difference is that the reason *Testing* is the second most common reason in publications complying with the DO-178 standard. An estimated 70% of all development costs for avionic systems are spent on testing, including software testing to a great extent [ISK+19]. Thus, the relative increase in occurrences in this category is well justifiable. Peleska's work [Pel18] focused on automatic model-based testing for verification and HW/SW integration testing of avionic controllers. The main advantage is the automatic identification of test cases in the system models, allowing for autonomous test data calculation, test procedure generation, and automated requirements tracing. The authors of [PE19] created a requirement modeling DSL for, inter alia, requirements-based testing. The DSL is designed as a UML profile and extends the SysML language. The profile includes three stereotype categories: firstly, requirement hierarchy, secondly, requirement interrelationship, and thirdly, requirement formalization. It can be used to generate test cases. Holzer et al. [HJK+11] developed their approach for model-based testing. The developed testing environment generates an automatically concretized test suite from a model. Their test process follows the three steps test generation, test concretization, and test evaluation. Furthermore, their approach can provide a solution for the DO-178 demand, for tests generated from requirements-derived models and seamless traceability of low-level requirements.

Eight publications mentioned the reason *Communication* and five publications the reason *Collaboration*. Both of these reasons occurred significantly more often than in other safety standards, in which they only occurred a few times. In [ASLN], experience and

lessons learned are presented for the model-based development of aviation components. The authors stated that models can simplify communications between parties involved in the design process. The reasons for this are the simplistic, formal and unambiguous representations of models.
An experimental evaluation of the understanding of safety compliance needs with models compared the understandability of textual descriptions and conceptual models [dlVMAG17]. Although their results were not conclusive enough, they showed that models improved the understandability of safety compliance needs by 17% in their experiment. Thus, we conclude that models could improve the communication between parties when it comes to safety requirements and their fulfillment criteria.
Blooshi et al. [BJ18] stated that models improved the stakeholder communication in their model-based development certification process. They argued that models eliminated implementation details that are irrelevant to the systems' logical behavior and reduced the language gap between the stakeholders and engineers. Thus, stakeholders are more involved in systems development. High involvement of stakeholders increased the probability of delivering a product that meets the business objectives. The authors of [WCEK17] presented model-based systems engineering as a solution for collaborative working in a systems engineering process, involving experts from several domains such as safety, security, and reliability.

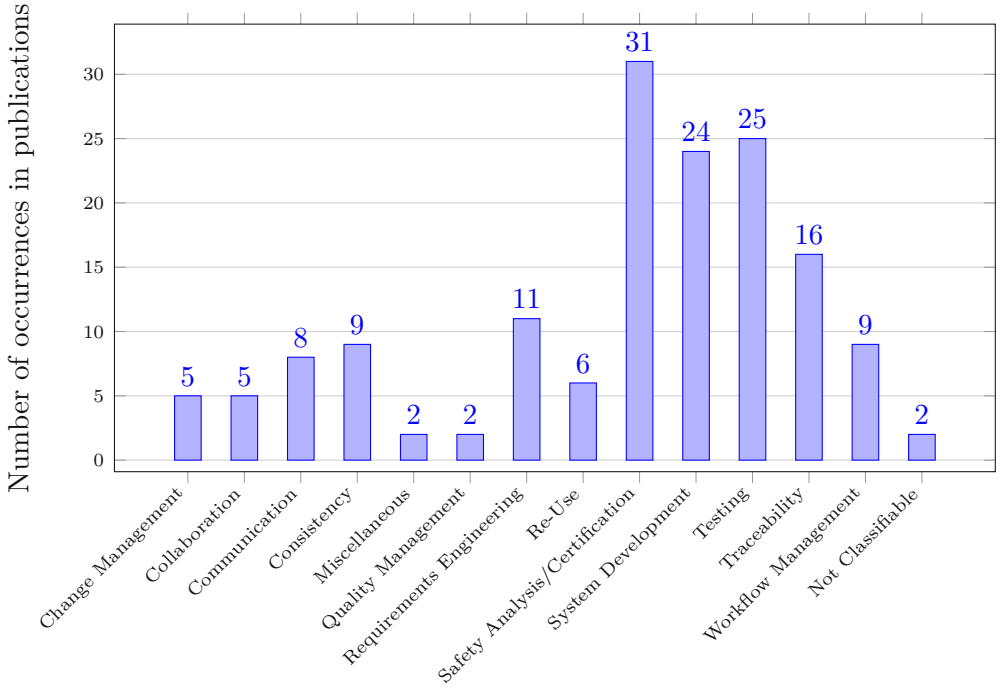

Figure 3.14: RQ2: Reasons for applying model-based methods in the context of DO-178 (Avionic).

EN 50128: For the railway safety standard EN 50128, only 23 publications were included for categorization during systematic mapping. Thus, the standard for software for railway control and protection systems is the standard with the fewest publications in the final result set. Figure 3.15 illustrates the number of stated reasons for applying model-based methods in the EN 50128 specific publications.

As with the generic IEC 61508 and the other two safety standards included in the SMS, the reason *Safety Analysis/Certification* is most common. Gallina et al. [GGME16] presented a modeling-approach based on the model-based MBASafe methodology and Model-Driven Safety Certification (MDSafeCer) for deriving EN 50128 compliant safety cases. The authors of [CAGUM18] stated that manual compliance with standards can be time-consuming and error-prone. They presented a prototype for transforming Systems & Software Process Engineering Meta-model (SPEM 2.0) models into compliance checkable models. A compliance checking use case comprised a software process model from the rail sector and results showed a possible increase in efficiency. Belmonte & Soubiran [BS12] introduced a transformation chain to generate domain-specific models for railway signaling systems. The generated models are dysfunctional models of the system. The domain-specific dysfunctional models allow the computation of fault tree analysis (FTA) and simulation of dysfunctional events leading to accidents, resulting in higher productivity and efficiency of safety engineers' study.

System Development was the second most common reasons, which matches the IEC 61508 results. In [SUN15], a modeling framework and pattern is introduced for the modeling and verification of French railway interlocking systems (RIS). The model transformations presented in the second part of the work can furthermore assist system designers from the analysis process till the final implementation. Hamid et al. [HGZG12] proposed a safety-oriented process meta-model for the systems development process. They are tackling the problem of integrating safety during process design by concentrating on the aspect of the safety life cycle.

The application of model-based methods for *Testing* is the third most common reason in the relevant publications addressing the EN 50128 safety standard. The high number of testing approaches matches the popularity of model-based testing for DO-178 avionics safety compliance to a certain extent. In [BDMM+14], an interoperable testing environment supporting the system level testing of railway control systems is described. The testing environment features integration testing between subsystems. Test cases are generated from a set of test specifications and transformed into executable test scripts, following a model-driven methodology. The support for integration testing between subsystems is important because subsystems are usually developed by different suppliers.

All other reasons were mentioned only four or fewer times and their publications are not discussed in more detail.

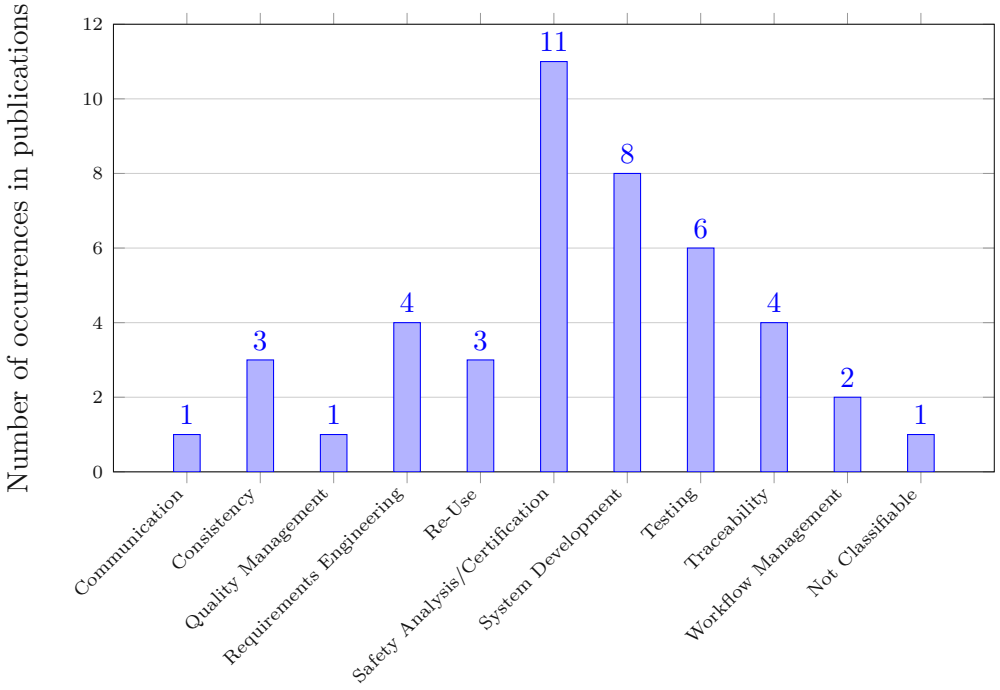

Figure 3.15: RQ2: Reasons for applying model-based methods in the context of EN 50128 (Railway).

In the previous bar charts (Figure 3.12-Figure 3.15) we showed the popularity of our defined thirteen reasons for applying model-based methods for each standard individually. To summarize the results of the second research question, the results of all four investigated standards are shown in Figure 3.16. In order to improve comparability, the results were converted from absolute numbers into percentages on the y-axis. For instance, 32% of all categorized DO-178 publications in the final result set mentioned the reason *Testing*. Compared to this, only 14% of publications addressing the ISO 26262 standard pointed out this reason. However, there is only an absolute difference of 5 occurrences between these two standards for the reason *Testing*.

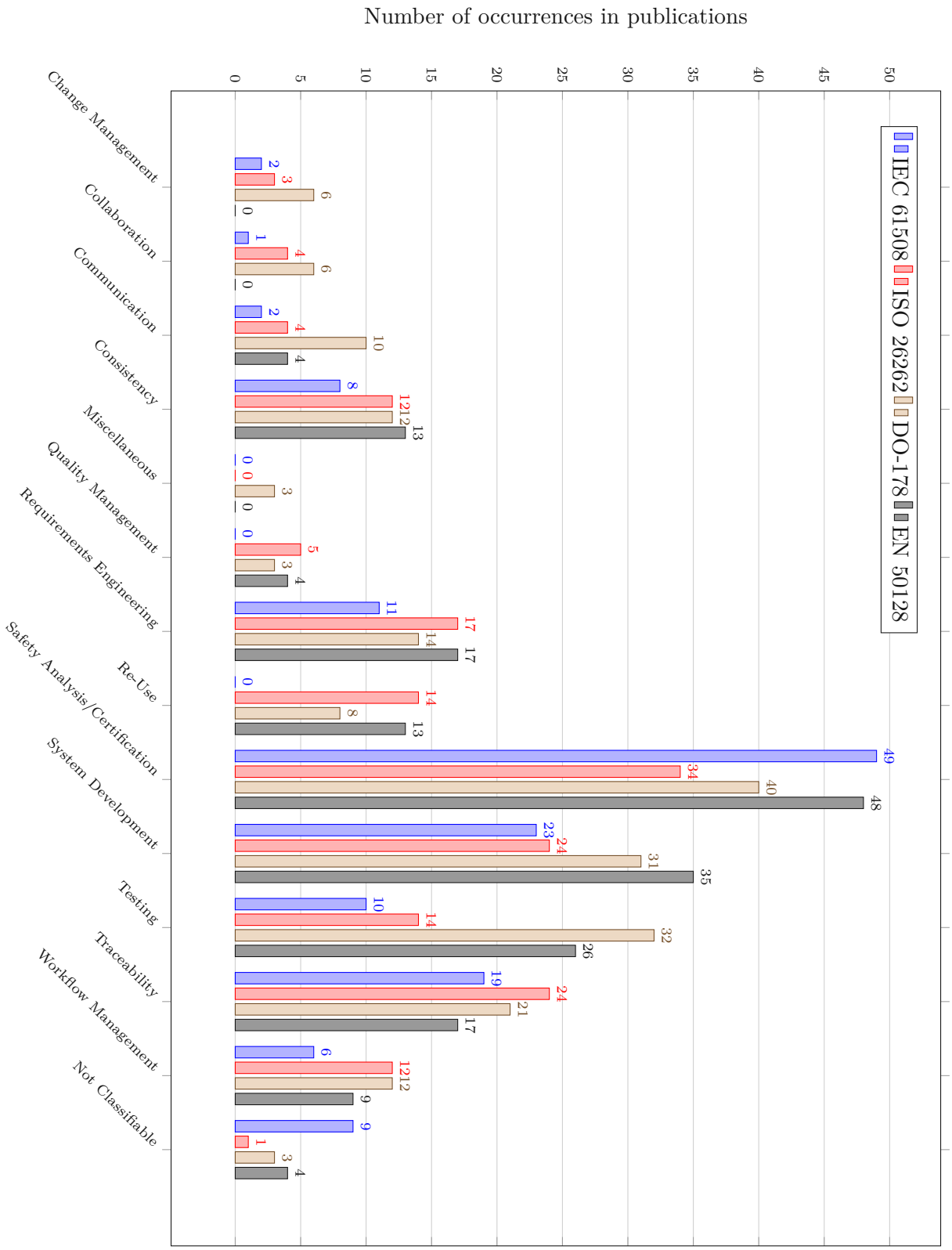

Figure 3.16: RQ2: Reasons for applying model-based methods (in %).

3.2.5.3 RQ 3: Which model-based methods are applied in IEC 61508 and each of the top three most common safety-critical standards in RQ 1?

Additionally to the categorization of publications based on the reasons for applying model-based methods investigated in RQ 2, this section discusses the insights into the applied model-based methods across the safety standards IEC 61508, ISO 26262, DO-178, and EN 50128 in detail. The categorization process of the third research question is similar to the process of the second research question, besides its categories. In contrast to RQ 2, the systematic mapping process was carried out based on the categories for model-based methods defined in the section 3.2.4. The reason for this is that we are not only interested in the reasons for applying model-based methods but also in the question which methods are applied in the publications. During the entire process to answer the third research question, multiple categorizations for each publication are possible, since more than one model-based method can be applied at a time.

IEC 61508: In a first step, we analyzed the model-based methods applied for the IEC 61508 safety standard. In total, 104 publications from the final result set of relevant publications were categorized. The results for the IEC 61508 standard are illustrated in Figure 3.17.

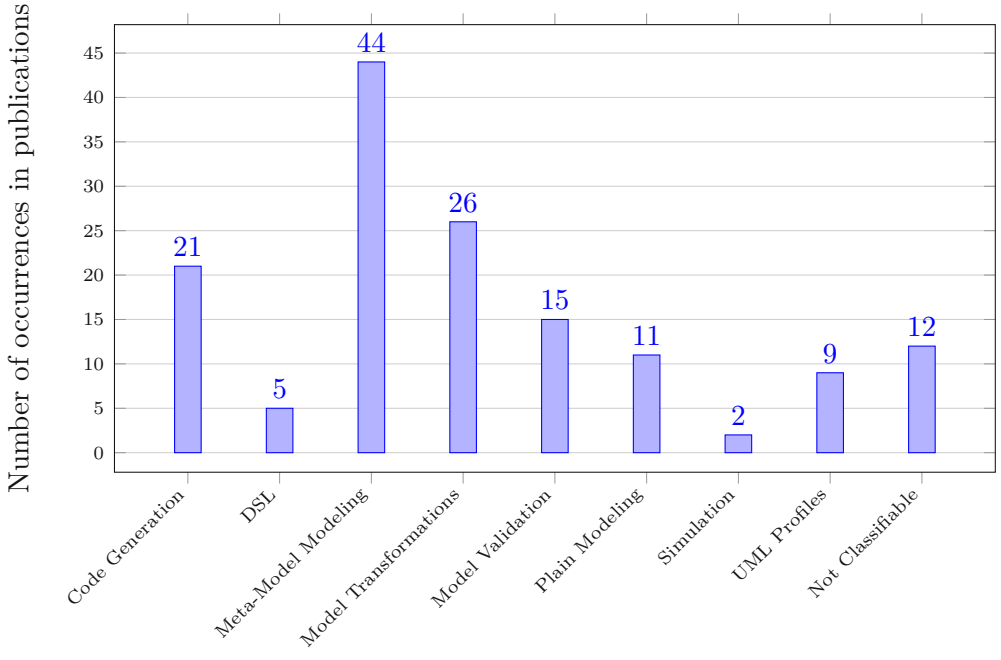

Figure 3.17: RQ3: Model-based methods applied in the context of IEC 61508 (Generic).

The model-based method *Meta-Model Modeling* was categorized 44 times and is the most common method for publications addressing IEC 61508. The popularity can, therefore,

be related to the creation or use of safety meta-models for safety analysis or safety certification to ensure safety standard compliance. Vara & Panesar-Walawege presented [dlVPW13] SafetyMet, a meta-model for facilitating safety compliance targeting different safety standards such as IEC 61508. Their approach comprises safety compliance models including project-specific information and the models of the safety standards that conform to the generic meta-model for safety standards (SafetyMet). Complex systems must sometimes be certified to different standards rather than be certified to just one standard. The major advantage of the SafetyMet methodology is that the meta-model is generic and flexible. Thus, it can demonstrate compliance with several safety standards for complex systems at once. In [SKB11], a standard-aware meta-model for using the fault models from IEC 61508 in model-driven development was developed. The work proposed that the use of a standard-aware meta-model minimizes the required knowledge for the development of standard-compliant safety-critical systems. The reason for this is the shift of knowledge from the developer into the modeling environment. Mayr et al. [MPS14] proposed an approach in the context of IEC 61508 for automatic quality assessment of software developed for safety-critical systems. The authors enhanced a meta-model to capture the requirements for the assessment to detect problems or check coding requirements. As the references discussed above have shown, *Meta-Model Modeling* has a wide range of applications.

Model Transformations is the second most commonly applied model-based method and was categorized 26 times in the publications. Vepsäläinen & Kuikka [VK13] applied model transformations from UML Automation Profile (AP) models to Modelica ML models for the simulation of safety-related interlocks. UML AP models are used for specification and are not applicable to simulation. The reason for the proposed model transformation is to enable support model-driven development by the automated creation of simulatable models.
In many safety-critical standards, such as the IEC 61508, fault tree models and fault tree analysis (FTA) are used as a safety evaluation method. Fault tree models are created based on system models, and changes in the system models must be incorporated in the fault tree models to do a complete evaluation after changes. The authors of [GGvH+18] designed model transformation rules to provide a semi-automatic co-evolution process between architecture software models and fault tree models. The work's results showed that the model transformations support developers and led to a significant reduction of the number of required user interactions to realize architecture software models and fault tree models co-evolution.

Code Generation is a model-based method similar to *Model Transformations* and these two methods are used quite often together. Six publications in the result set have used both methods. In [Con09], a model-based design workflow was discussed. The workflow included IEC 61508 compliant verified and validated *Code Generation*. The advantage of such a workflow is a decrease in risk due to the reduction of errors during development. Buckl et al. [BSK10] also stated that model-driven software development

and code generation reduce the probability that developers introduce software design faults to a certain extent. Their presented model-driven development tool for developing fault-tolerant real-time systems is called FTOS. FTOS provides *Code Generation* and formal methods to verify the generated code.

UML Profiles and OCL constraints for *Model Validation* were applied in a model-driven engineering approach by Panesar-Walawege et al. [PWSB11]. Their work showed the automated verification of compliance to evidence requirements in the IEC 61508 standard.

Further publications addressing other model-based methods are not discussed in detail for the IEC 61508 standard.

ISO 26262: The categorization process for the ISO 26262 safety standard, entitled *Road vehicles - Functional safety*, included the highest number of publications of all standards. 147 publications were analyzed for their applied model-based methods in the systematic mapping process. In contrast to the IEC 61508, the method *Model Transformations* is applied more often than the method *Meta-Model Modeling*. Furthermore, the method *Model Transformations* is the most commonly applied model-based method in the ISO 26262 final result set. Surprisingly, *Plain Modeling* is rather popular and occurred 37 times. Furthermore, this model-based method is most commonly used compared to the other standards. *DSL* and *Simulation* became quite a bit more popular compared to IEC 61508 and are applied in several publications.

Model Transformation is a model-based method applied in almost every second ISO 26262 publication. In [LKB19], conceptual models created in the conceptual phase were transformed into specification models for safety case assessment to reduce manual work involved in the development. They stated that manual work for safety assurance is time-consuming and expensive. Furthermore, *Model Transformations* can support safety assurance when a system evolves and needs to be regathered or revalidated because of automated processes rather than manual work. Gallina [Gal14] introduced a model-driven safety certification method for process compliance in the context of ISO 26262. Automatic *Model Transformations* generate process-based argument models from process models. Process-based arguments are necessary to show that a development process matches the requirements according to the applied standard. The automatic generation enhances re-use and reduces time and cost. Maro et al. [MSS18] mentioned that *Model Transformations* can support the automated generation of traceability links. However, in the automotive industry not all artifacts are necessarily models. This can make it difficult to integrate the artifacts into the transformations to support traceability links. In [MAT+18], safety artifacts for FMEA and FTA are automatically generated from extended system models by *Model Transformations*. This process has several advantages for safety analysis and optimization. Some of the advantages are that the effects of changes in the system models are directly visible in the generated artifacts and the safety assessment of the system picks up the pace.

The model-based method *Meta-Model Modeling* is the second most stated method in the ISO 26262 publications. In Durisic's work [SD17], the Automotive Open System Architecture (AUTOSAR) meta-model is explained based on a development process. A worldwide standard for the development of software architecture such as AUTOSAR for automotive software architecture eases the exchange of models between different parties and at different stages in the development process because every model conforms to the same meta-model.
Conceptual models for safety-critical systems are created based on domain-specific meta-models. Luo et al. [LEvdB14] stated that conceptual models in the safety-critical domain may be created in the form of meta-models using different concepts from overlapping domains. For instance, an IEC 61508 meta-model can be refined into an ISO 26262 meta-model when sufficient mapping information is available. Their work focused on the detected mappings when meta-models and models used for safety assurance are compared. The mappings are necessary to facilitate linkage between (meta-)models and their refined (meta-)models. The authors of [ADTK12] created an extended meta-model to allow the description of the two standards ISO 26262 and HIS in one model. This meta-model allows for a full assessment of compliance with both standards. The goal of the proposed methodology is to facilitate an acceptable certification perspective for simultaneously performed HIS assessment and ISO 26262 functional safety audit.

The use of *Plain Modeling* is quite popular compared to the analyzed model-based methods applied in publications addressing one of the other three standards. *Plain Modeling* is usually applied with the Systems Modeling Language (SysML) or Matlab/Simulink and combined with other approaches because the applications are limited. Baumgart et al. [BFP15] researched product lines approaches and stated that they are not sufficient for tracing safety-related information through the development process. In their work a model-based approach is described to capture safety-related characteristics in the concept phase. UML use case diagrams are used to capture the relation between the systems functionality and the users and add safety-related information to application scenarios during the concept phase. Bonfiglio et al. [BMR+15] focused on software safety analysis in the context of ISO 26262 by model execution and fault injection at the model level. The paper describes a process of obtaining an executable model from a component-based UML model by adding information with UML, MARTE and fUML modeling constructs. The executable model is the basis for performing software failure mode and effects analysis (FMEA) and fault injection, as recommended in the ISO 26262 specifications. In [Alt14a], the author stated that functional safety management has to manage rising complexity to satisfy ISO 26262 requirements for systems under development. Model-based development with UML and SysML is a model-based method able to minimize or solve these problems. A further advantage is that UML and SysML are extendable languages and enable the integration of specific functional safety management aspects [Alt12].

In the automotive domain, the EAST-ADL was mentioned in several publications such as [AGMG11, CJL+11, MAL+11, Sta17, Pie19, ZTK18]. EAST-ADL provides a well-

defined information structure for the development of automotive systems and is suitable for architecture specification, safety requirements, safety constraint, fault modeling, and more. More detailed information can be found in [BCK+16].

Simulation facilitates the analysis of correct behavior in the early stages of development. For instance, some approaches in the categorized papers are using fault injection at the model-in-the-loop (MiL) level. Some simulation technologies are also applicable to fault injection at the hardware-in-the-loop (HiL) level [MVAVD18, NBR+19, MLL+17]. The increasing popularity of simulation frameworks can be attributed to the fact that the behavior of self-driving vehicles is imitated to cope with the overwhelming test effort [GKR+17].

Since the ISO 26262 result set included 147 publications, it is not possible to discuss the majority of them in detail, even if most of them pursue interesting and widely distributed research goals for model-based approaches.

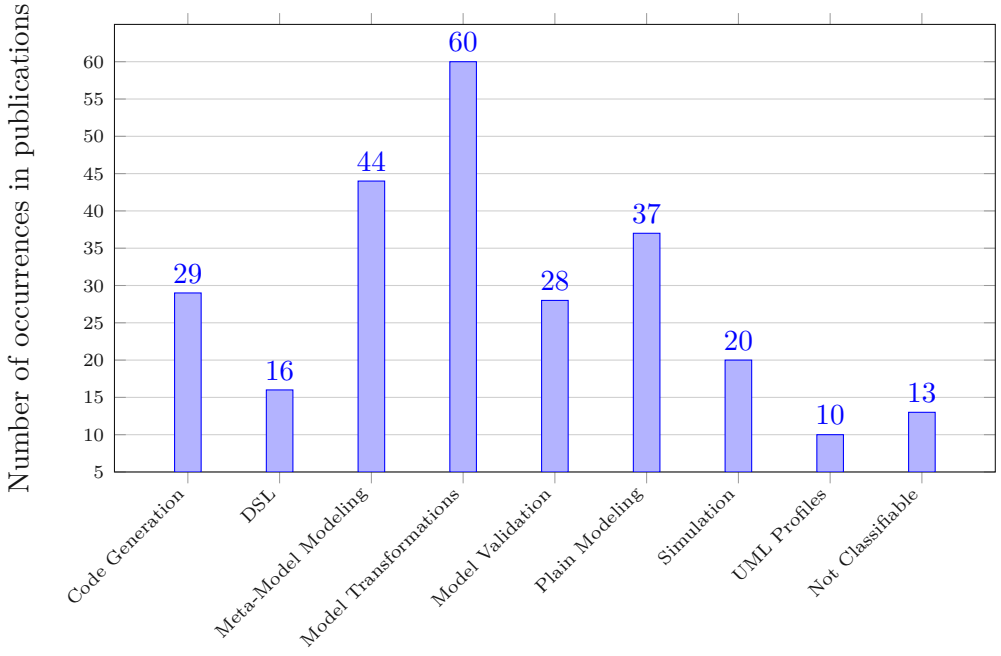

Figure 3.18: RQ3: Model-based methods applied in the context of ISO 26262 (Automotive).

DO-178: In the final result set, 77 publications addressing the DO-178 standard for Software Considerations in Airborne Systems and Equipment Certification were analyzed for their applied model-based methods. The categorization process for the reasons of applying model-based methods showed that the reasons *Safety Analysis/Certification* and *Testing* are the two common reasons justified. This confirms the result that the model-based

method *Model Transformations* is the most applied one and the method *Code Generation* is the second most applied one. Model-based testing is mostly dependent on automatic generation of test artifacts, and the model-based methods *Model Transformations* & *Code Generation* can support this automatic generation to a great extent. *Meta-Model Modeling* was practiced 18 times and *DSL* 17 times. Thus, *Meta-Model Modeling* is not as popular as in publications addressing IEC 61508, ISO 26262, or EN 50128. The model-based method *DSL* has the highest popularity in avionic publications as well as in automotive ones. All other model-based methods were used on average often, compared to the other safety standards. All results of the categorization process for model-based methods applied in DO-178 are illustrated in Figure 3.19. The most interesting publications and their applied model-based methods in the final result set are discussed below.

Model Transformations were applied in 29 publications and, therefore, it is the most applied model-based method in the results set. The authors of [PDA17] stated that safety assurance and validation of embedded airborne systems is becoming more and more complex. Their work applied *Model Transformations* to automate the generation of fault tree models and the generation of comprehensive test cases for safety validation at the system level. A use case showed that the model-based generated test cases are more effective than the ones generated traditionally. However, a major weakness is that the automatic generation is only as good as the model. Therefore, a bad model results in weak testing. However, a bad model can be tested well with manual testing approaches. Grant & Datta [GD15] presented a DO-178C conform model-based software engineering methodology focusing on formal specification techniques to incorporate standard compliance. The specification language used to conduct formal analyses was the Z notation [Spi89]. *Model Transformations* transform informal UML models into formal Z models to enable this formal analysis. Errors discovered during formal analysis are corrected in the formal model, and repeated correction-transformation cycles are executed until an acceptable level of safety assurance is achieved in the context of DO-178C. Mkaouar et al. [MZHJ19] presented a similar but more generic methodology for the use of formal verification in a model-based development process. The system architecture is modeled in Architecture Analysis & Design Language (AADL), a model-based engineering language for embedded real-time systems. AADL models are transformed into LNT specifications by *Model Transformations*. LNT is a formal specification language and allows for the formal verification and specification of real-time features. A flight controller development was discussed in their work as an example.

Schuhmann & Goseva-Popstojanova [SG19] presented a verification & validation (V&V) architecture specifically designed for model-based software engineering and the use of automatic *Code Generation*. Two case studies are discussed based on two NASA space missions and showed that model-based software engineering could be successfully used for safety-critical avionics systems. In the first project approximately 18% of the mission code was auto-generated, and in the second project approximately 35% of the mission code was auto-generated. In both projects no manual modification in the code was

necessary. All changes were made in the model elements and synchronized in the code with the next build. This shows that the method *Code Generation* can already cover a great many requirements.
In [PK15], a model-based solution for the development of Unmanned Air Vehicles (UAV) is discussed. C program code was automatically and directly generated from SCADE models (software architecture and design models). Their approach used qualified code generation and can significantly reduce development time and speed up time-to-certification. As demonstrated in [PK15, p. 4], "Application of the mentioned software development technology allows considerably reduce its development and maintenance due to effective organization of the most labor-intensive process – verification. It allows reducing costs of all software life cycle no less than 40% as compared to the hand-coding."

The SPES 2020 research program worked on the improvement in model-based systems engineering, model-based software engineering and verification in the avionics domain. The developed methods helped to close existing gaps between systems and software modeling in the avionics domain and uses domain-specific modeling languages *(DSL)* [BHG+12].

Estrada et al. [ESD13] published best practices for developing DO-178 compliant software using model-based design. Among other things, they cited that *Simulation* should be used as a verification and validation technique to demonstrate compliance of design model artifacts with software requirements. Furthermore, the authors noted that "simulation can replace traditional reviews and analysis to provide reproducible evidence of compliance with a subset of objectives for high and low-level requirements" [ESD13, p. 9]. For instance, tools such as Simulink Report Generator can be used to automatically capture simulation results and simplify manual review.

Zoughbi et al. [ZBL11] developed a *UML profile* to improve communication and collaboration among safety engineers, software engineers, and certification authorities in the context of DO-178B. For instance, it allows to monitor the implementation of safety requirements and supports system certification. Furthermore, their methodology is able to automatically generate reports containing safety and certification-related model information to facilitate the certification process between system developers and certification authorities. The reports provide a good basis of evidence for the fulfillment of DO-178B development requirements. The developed *UML profile*, entitled SafeUML is the only *UML profile* approach of this kind in all of our 351 categorized publications.

As in the discussion of the applied model-based methods in the IEC 61508 and ISO 26262 standard, it is not possible to discuss all publications in detail. In this subsection, the most interesting publications in the context of DO-178 were presented.

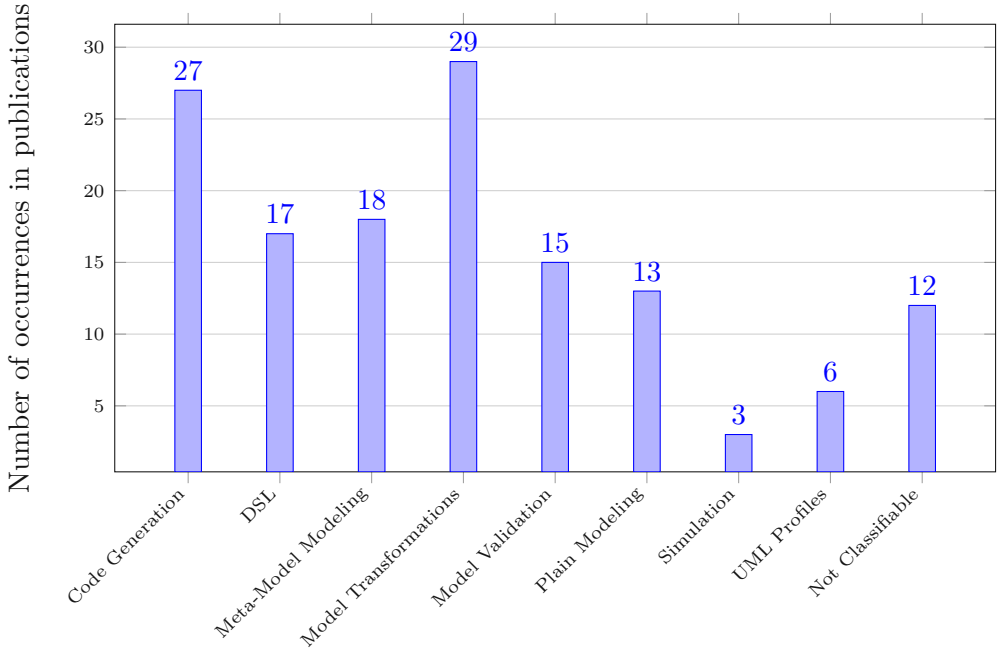

Figure 3.19: RQ3: Model-based methods applied in the context of DO-178 (Avionic).

EN 50128: The functional safety standard EN 50128 for railway applications is the standard with the least publications of all four examined standards. Only 23 publications were included for analyzing the model-based methods applied in the railway domain. The results of the classification process in the context of the EN 50128 safety standard are illustrated in Figure 3.20. Statistically speaking, this set is not very meaningful due to the small number of publications the result set contains. However, it can provide evidence about the ongoing research in model-based engineering in the railway domain.

As with the ISO 26262 and DO-178 standards, *Model Transformations* were used most frequently and occurred a total of 15 times in the categorized publications. The most obvious difference is that *Model Validations* is the second most frequently applied model-based method and occurred 11 times. In the other three standards, the model-based method *Model Validations* was not even in the top three of the most common ones. The results for the other model-based methods showed no significant differences compared to the previous investigation results. In the following paragraphs some of the most important publications from the final result set are discussed.

Gallina et al. [GGME16] showed that the model-based design methodology for Assessing performance and safety requirements of critical systems (MBA[10]) is partly compliant

[10]MBA is a model-based methodology for the verification of safety contracts and performance requirements in the design phase [GMRBE+18].

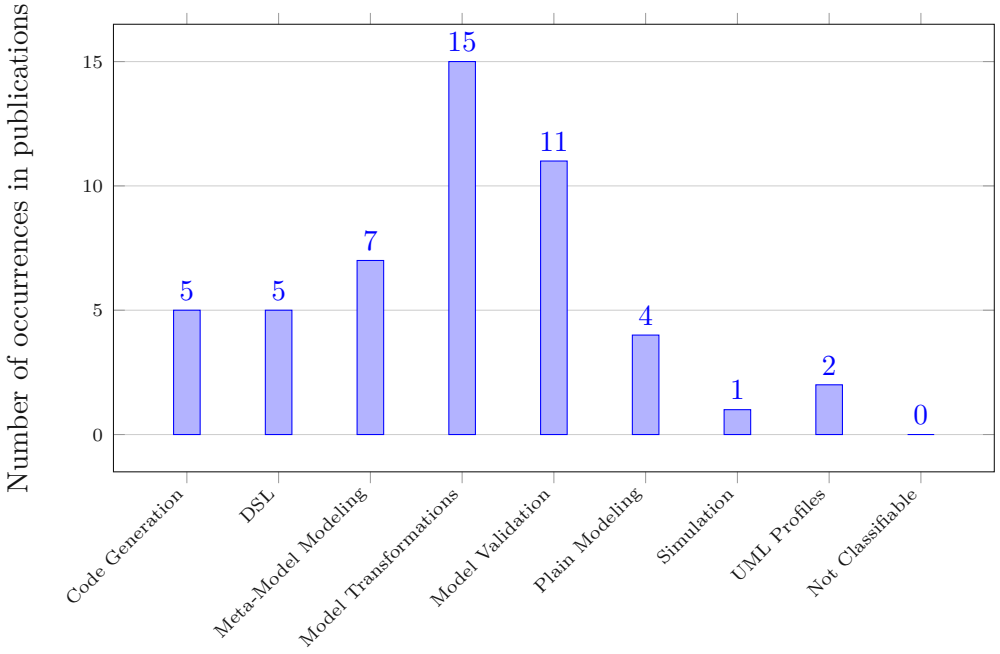

Figure 3.20: RQ3: Model-based methods applied in the context of EN 50128 (Railway).

with EN 50128. In order to obtain the necessary formal architectural specification, *Model Transformations* are needed. Due to the complexity the transformations are carried out in two steps. Firstly, MARTE annotated UML diagrams are transformed into Generalized Stochastic Petri nets (GSPN) by automatically applied *Model Transformations*. Secondly, Object Constraint Language (OCL) constraints are manually converted based on specific guidelines. An automatic transformation of OCL constraints has not been accomplished in their work yet. The authors of [LvdBEK14] presented a *Meta-Model Transformation* approach to derive safety-standard or project-specific meta-models from generic meta-models to facilitate safety assurance. Existing certification data can be re-used when certain certifications from the generic meta-model also apply to the specific meta-model. The goal is to reduce the high costs of (re-)certification. With regard to this thesis, this means that an IEC 61508 conceptual model could be refined by *Model Transformations* into an ISO 26262 or EN 50128 conceptual model (see Figure 3.21).

In [SOE+08], *Model Validations* were applied to ensure that the systems conform to the requirements. All created OCL expressions need to be fulfilled for a valid model. For instance, a constraint that only certain kind of railway signals can be placed at certain locations is mentioned by the authors. Sango et al. [SDG15] manually transformed component models to Uppaal Timed Automata (TA) models to validate temporal safety requirements. Uppaal is an automatic model checker tool [LPY97]. Hamid et al. [HGZG12] proposed a safety-oriented process meta-model to support all the requirements of safety processes in the example of the railway domain. They furthermore compared

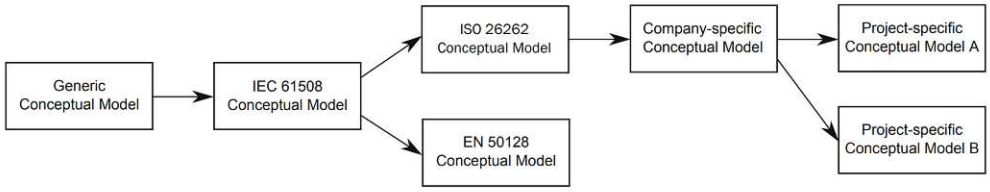

Figure 3.21: The successive refinement of meta-models creates a set of closely related meta-models ([LvdBEK14, p. 203]).

the validation suitability of the existing process meta-models Software & Systems Process Engineering Meta-model (SPEM), Unified Method Architecture (UMA), and OPEN Process Framework (OPF). UMA and OPF are better suited for validation tasks than SPEM. The authors of [WLG09] stated that formal verification minimizes the risk of discovering errors during the validation phases. Thus, *Model Validation* should not be used to detect specification errors.

The previous Figures (Fig. 3.17-3.20) showed the popularity of applied model-based methods based on the eight categories defined in the section 3.2.4. In total, four bar charts were created to identify to what extent model-based methods are applied. Each standard was investigated individually. To summarize the results of the third research question, all the results of RQ 3 were combined and are illustrated in Figure 3.22. In the same way as for the final results of RQ 2, we converted the results from absolute numbers into percentages on the y-axis to allow for a comparison of the values. For instance, 26 publications in the IEC 61508 result set applied *Model Transformations* in their work, but only 15 publications addressing the EN 50128 standard applied this model-based method. After converting the absolute numbers to percentages, only 25% of all IEC 61508 publications applied *Model Transformations*, but 65% of all DO-178 publications did.

A summary of the Systematic Mapping Study and the results of the three investigated research questions is discussed in the following section.

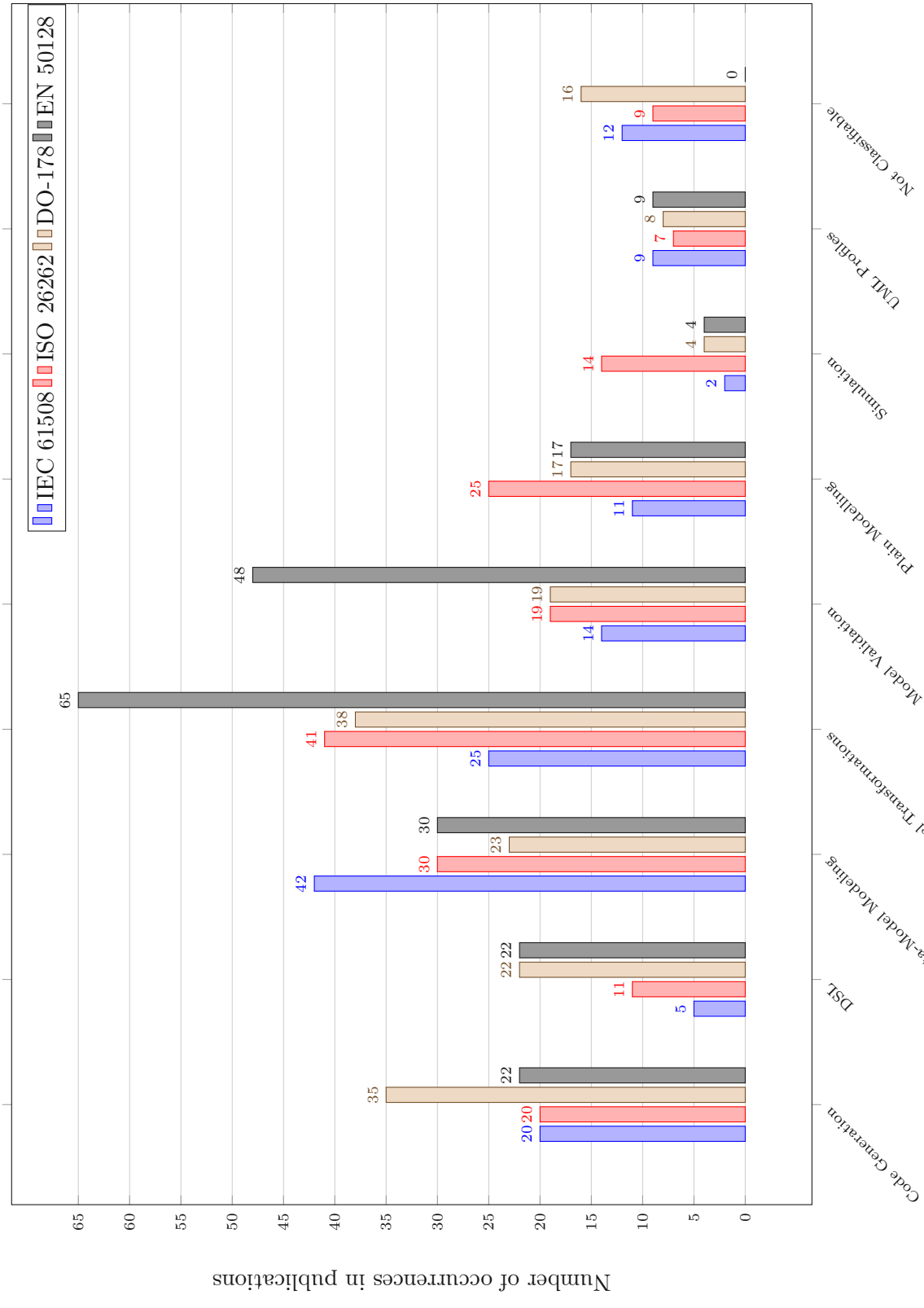

Figure 3.22: RQ3: Applied model-based methods (in %).

3.3 Summary

In this part of the thesis, an SMS has been conducted for investigating model-based engineering for the development of security-certified safety-critical systems. The contribution of this SMS can be structured into the following three parts:

1. Model-based methods are by far the most frequently applied in the automotive domain, followed by the avionic and railway domain. This is also consistent with the most frequently addressed safety standards ISO 26262 (Automotive), IEC 61508 (Generic), DO-178 (Avionic), and EN 50128 (Railway).

2. The reasons for applying model-based methods are manifold and slightly different between the four safety standards for functional-safety. Nevertheless, in summary, the reasons *Safety Analysis/Certification*, *System Development*, and *Traceability* were the most commonly stated reasons for applying model-based methods.

3. The characteristics of the applied model-based methods are almost identical with the reasons for using model-based methods. There are some differences in the methods applied between the different safety standards, but on the whole the methods applied and their popularity are rather similar. The most commonly applied model-based methods are *Model Transformations*, *Meta-Model Modeling* and *Code Generation*.

In summary, the results have shown that model-based systems engineering in the context of the four investigated standards IEC 61508, ISO 26262, DO-178, and EN 50128 mostly concerns safety assurance. The impression is, however, that many solutions proposed are not yet fully developed. In fact, Vara et al. share this impression and note:

> *"Finally, and based on our knowledge and experience, the full adoption of model-driven engineering for assurance of safety-critical systems needs to overcome some barriers. Challenges arising from practical aspects such as scalability, efficient model storage, and tool qualification must be tackled, at least for many open-source solutions. From a research perspective, the development of model-driven engineering solutions that cover a wide range of domains and of dependability concerns remains an area where further work is necessary"* [VRE18, p. 639].

All data of the results and systematic mapping can be found at: `https://www.dropbox.com/sh/4eep2gb9rtsumgd/AABVRqQBYIRYJ93k4vZj05DWa?dl=0/`.

CHAPTER 4

SysML Extension for ASIL Tailoring and Decomposition

In this chapter, the systematic development of the DSL, entitled *ASIL SysML Profile*, is described both theoretically and practically. The DSL consists of a SysML profile and OCL statements to define an appropriate means for the ASIL tailoring and ASIL decomposition concepts in the SysML. These concepts are both also described in detail. Furthermore, the DSL must ensure fully compliant model-based systems engineering in the context of several ISO 26262 clauses. In order to ensure this ISO 26262 compliance, twenty-two validation criteria are defined and evaluated. This descriptive evaluation shows the benefit of the DSL to use SysML for modeling ASIL tailoring and ASIL decomposition. Afterward, a use case is presented to show a real-life model-based systems engineering example based on the development of an ISO 26262 compliant car dashboard. The entire DSL development process follows the design-science paradigm from [HRM+04] and the approach for the systematic development of domain-specific languages from [SZ09]. The technology used for implementation is the Model Driven Generation (MDG) technology from [Spaa]. Both scientific guidelines and the MDG technology are described in more detail in this chapter.

4.1 Introduction and Background

The Automotive Safety Integrity Level (ASIL) specified in the ISO 26262 series of standards is a complex and not yet thoroughly researched concept. Since the ISO 26262 provides non-prescriptive guidance, the ASIL concept requires a certain amount of interpretation by the user. There are several ways to deal with the high number of requirements to ensure ISO 26262 compliance during the systems development process. Rapid technological progress like self-driving cars furthermore requires new development methods for safety assurance for road vehicles engineering. A new approach to disburden

the sophisticated task of safety assurance is the application of model-based systems engineering. Model-based approaches are intended to replace document-centric approaches because of many different reasons and advantages. The Systematic Mapping Study in chapter 3 discussed all of them in detail. However, so far only few papers have been published that deal with an appropriate UML-based notation to facilitate ISO 26262 compliant model-based systems engineering. These papers are discussed in subsection 4.1.1. Most importantly, none of them has focused on the ASIL decomposition concept in detail yet. This work addresses this question of an appropriate means to represent the ASIL decomposition concept in a DSL. In the following subsections, related work in this area of research is introduced. Furthermore, the ninth part of the ISO 26262, entitled *Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses*, is described to provide the necessary knowledge for applying ASIL decomposition.

4.1.1 Related Work

In the previous chapter 3, a large number of publications addressing ISO 26262 compliant model-based systems engineering approaches have already been discussed. This subsection presents related work with a great interest in our SysML ASIL decomposition DSL.

Habli et al. [HIRK10] were one of the first to examine model-driven development for justifying automotive functional safety. Their work even observed a draft of the automotive functional safety standard ISO 26262 in 2010 rather than the final release of the following year. Safety cases are explicitly required for ISO 26262 compliance. The authors focused on the systematic and model-driven generation of functional safety requirements considered in safety cases. Their model-driven approach represents safety cases in the Goal Structuring Notation (GSN) and system models in the SysML. A safety case presents an argument, supported by evidence, that the system is acceptably safe to operate in a given environment. The graphical presentation of safety arguments had major advantages compared to descriptions in free text. In the next step, the safety arguments modeled in GSN were traced to the system models, modeled in SysML. The GSN model contains information about acceptable safety behavior just like ASIL. In this way the SysML models are linked with automotive safety cases. Furthermore, in their approach safety cases could be structurally and traceably developed for ISO 26262 compliance.

Fockel [Foc16] researched ASIL tailoring for functional safety requirements. The goal of ASIL tailoring is to develop most subsystems with lower ASIL requirements. This is only possible if subsystems are separated or redundantly implemented. This decision for separation or redundancy is usually made late in development processes and can thus cause costly development iterations. The author presented a model-based systems engineering method for ASIL tailoring in the requirements analysis phase. The requirements analysis phase is a phase in the early stage of the overall development process of safety-critical systems. The advantages of an early planning of safety measures during the requirements analysis are time and cost savings by avoiding late development iterations.

Thus, this model-based approach based on SysML models may be able to tackle the safety requirements engineering dilemma. As seen in [Foc16, p. 12], "The safety requirements engineering dilemma states, that failures can best be found late in the development process but ideally would be found early, already in the requirements analysis phase. Because then, required safety measures and resulting effort and cost can be planned from the start."

Pétin et al. [PEML10] combined SysML and formal methods for safety requirements verification for safety-critical systems. Requirement and block diagrams are created in SysML and transformed to UPPAAL[1] using model transformations as recommended by MDE. UPPAAL is a model checker and facilitates model checking that allows, as stated in [PEML10, p. 9], "to prove that the local behavior of each system component contributes to satisfy system requirements".

However, the three presented related works do not develop a UML profile or SysML profile. There are only a few publications that have developed a UML profile for safety assurance for IEC 61508 and its application-specific variants. More precisely, their research goals primarily pursue model-based approaches for assuring functional safety in the context of ISO 26262 for road vehicles and DO-178 for airborne systems. Zoughbi et al. [ZBL11] developed a UML profile to model safety-related concepts and properties for the DO-178B standard. Their work has already been discussed in more detail in sub-subsection 3.2.5.3. For safety assurance according to the ISO 26262 functional safety standard the developed profiles, as discussed below, are more advanced than the profile for DO-178B.

In [BHFH13], a model-based Hazard Analysis and Risk Assessment method for automotive systems is presented. Hazard Analysis and Risk Assessment is required by ISO 26262 to determine unavoidable safety measures. Their developed UML profile, called *UML4PF*, can express all elements of a Hazard Analysis in compliance with the ISO 26262 safety standard. The UML profile furthermore supports validation conditions expressed as formal OCL expressions that enable validation concerning consistency and correctness and facilitate review activities required by ISO 26262. Specific stereotypes such as *SafetyRequirement, SafetyGoal, Fault, Hazard* and *RiskAssessment* are defined. The ASIL is included as separate attribute. In this thesis, the shift from document-centric approaches to model-based approaches has been mentioned several times. In fact, Wilde et al. [BHFH13, p. 247] noted that "hazard analysis in practice is currently table-based using spreadsheets like Microsoft Excel" as of 2013 but that model-based approaches require less effort than existing document-centric approaches. In [BCF+14], the authors extended their model-based approach by the possibility to break down the defined safety goals into newly defined functional safety requirements. The UML profile from [BHFH13] is extended to express the functional safety concept in compliance with ISO 26262 and defines stereotypes such as *FunctionalSafetyRequirement* and *DecompositionRequirement*. Added validation conditions ensure the consistency and correctness of the functional safety concept mapped in models. The extended UML profile facilitates safety reason-

[1] For more information about UPPAAL see [LPY97].

ing and requires less effort to manage functional safety management. In [BCF+15], a model-based development method is presented based on their two previous works and illustrated using an ISO 26262 certified three-wheeled-tilting control system as running example and case study. Furthermore, an interface to the suppliers is defined because the ISO 26262 standard also concerns automotive suppliers to a great extent. The proposed model-based development method has been applied successfully to several Ford of Europe projects.

Even though ASIL tailoring or ASIL decomposition have been mentioned a few times, there is no complete modeling concept for these two methods using a UML profile—an issue addressed by this chaper.

4.1.2 ISO 26262:9[2]

The ninth part of the ISO 26262 series of standards, entitled *Road vehicles — Functional safety — Part 9: Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses*, describes a framework for functional safety to assist the development of safety-related E/E systems that are installed in the series production of road vehicles (excluding mopeds). The state-of-the-art version of the ISO 26262 was released in 2018. This part specifies the requirements for ASIL-oriented and safety-oriented analyses, including the following [ISOm, p. 1]:

- requirements decomposition with respect to ASIL tailoring;
- criteria for coexistence of elements;
- analysis of dependent failures; and
- safety analyses.

This thesis focuses on the first item of the list, namely requirements decomposition with respect to ASIL tailoring. ASIL decomposition is an ASIL tailoring method applied during the concept and development phases. ASIL tailoring is an approach for the accurate assignment of a safety integrity level. The ASIL is defined as "one of four levels to specify the item's or element's necessary ISO 26262 requirements and safety measures to apply for avoiding an unreasonable risk, with D representing the most stringent and A the least stringent level" [ISOd, p. 2]. Furthermore, ASIL decomposition is specified by ISO 26262 as:

> *ASIL decomposition is a method of ASIL tailoring during the concept and development phases. During the safety requirements allocation process, benefit*

[2]This subsection extends section 2.2.2. Thus, the basic concepts of the ISO 26262 standard for functional safety are not explained in detail again.

can be obtained from architectural decisions including the existence of sufficient independent architectural elements. This offers the opportunity:

- *to implement safety requirements redundantly by these independent architectural elements, and*
- *to assign a potentially lower ASIL to (some of) these decomposed safety requirements.*

If the architectural elements are not sufficiently independent, then the redundant requirements and the architectural elements inherit the initial ASIL [ISOm, p. 4].

Figure 4.1 illustrates an example architecture according to [ISOm] and its clause 5 called *Requirements decomposition with respect to ASIL tailoring* and its clause 6 *Criteria for coexistence of elements.*

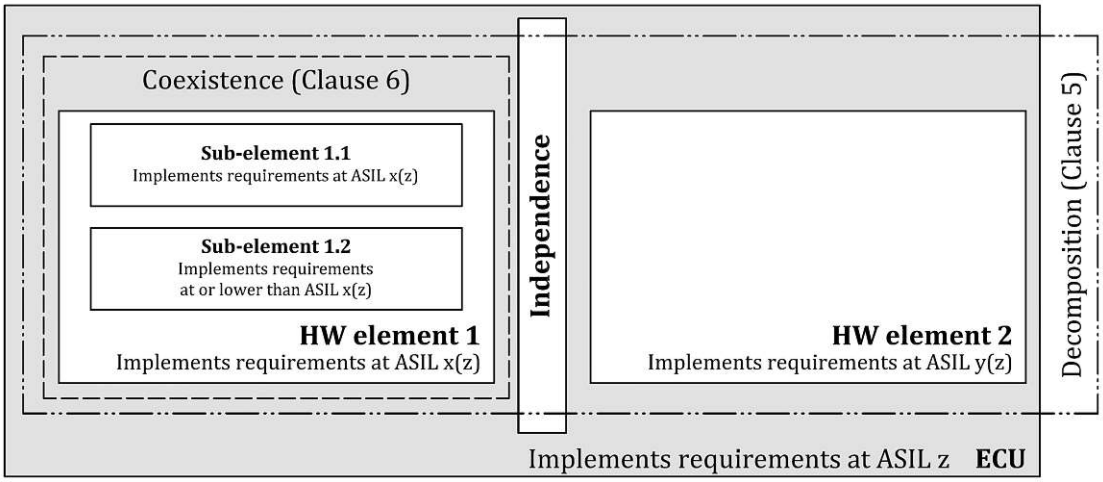

Figure 4.1: ISO 26262-9 coexistence and decomposition concepts in an example architecture [ISOm, p. 23].

This subsection is only intended to provide an overview of part 9 of the ISO 26262 series of standards. The detailed concepts relevant for the SysML profile and OCL constraints are explained in more detail in section 4.2, namely the design-science framework that serves as a guideline for the information systems research conducted in this chapter, and the safety life cycle concerning the concepts of ASIL tailoring and decomposition.

4.2 Theoretical Approach

In this section, the ISO 26262 safety life cycle concerning the concepts of ASIL tailoring and decomposition and the design-science framework are described in detail. The following subsection 4.2.1 provides all necessary theoretical background and definitions in order to implement an ISO 26262 compliant DSL as described in section 4.3. Furthermore, the information provided is of great importance for the descriptive evaluation performed in section 4.4. In subsection 4.2.2, the guidelines for design-science research within the discipline of information systems and its corresponding design-science research framework proposed by Hevner et al. [HRM+04] are presented. The design-science research framework has been adapted to fit the research conducted in this chapter.

4.2.1 Safety Life Cycle Concerning ASIL Tailoring and Decomposition

In this subsection, firstly, the safety life cycle and its management activities are briefly explained. Secondly, the exact workflow of ASIL tailoring and decomposition is explained in detail in the context of the ISO 26262 parts concerned.

Figure 4.2 illustrates the management activities in relation to the ISO 26262 safety life cycle. The key safety management tasks can be divided into three parts. Firstly, the overall safety management (see [ISOf, p. 4-12]), secondly, the project-dependent safety management (see [ISOf, p. 12-28]) and thirdly, the safety management regarding production, operation, service and decommissioning (see [ISOf, p. 28f]). The activities in which ASIL decomposition can be applied are marked in red. The phases and subphases of the safety life cycle are introduced below. We will focus on the phases and subphases in which ASIL tailoring and decomposition are of major importance:

(a) Item definition: The item, its functionality, dependencies on, and interaction with, the driver, the environment and other items at the vehicle level are defined and described. The definition and description must support adequate understanding so that subsequent activities in the life cycle can be performed [ISOg, p. 83f].

(b) Impact analysis at the item level: This activity at the beginning of the ISO 26262 safety life cycle determines whether the item is a new development, a modification of an existing item or an existing item with a modified environment. As for the modifications, a distinction is made between modifications to the design, modifications to the implementation, and modifications related to the environment, all of which are analyzed for their influence on functional safety [ISOf, p. 15].

(c) Hazard analysis and risk assessment: The hazard analysis and risk assessment identifies and classifies hazardous events caused by malfunctioning behavior of the

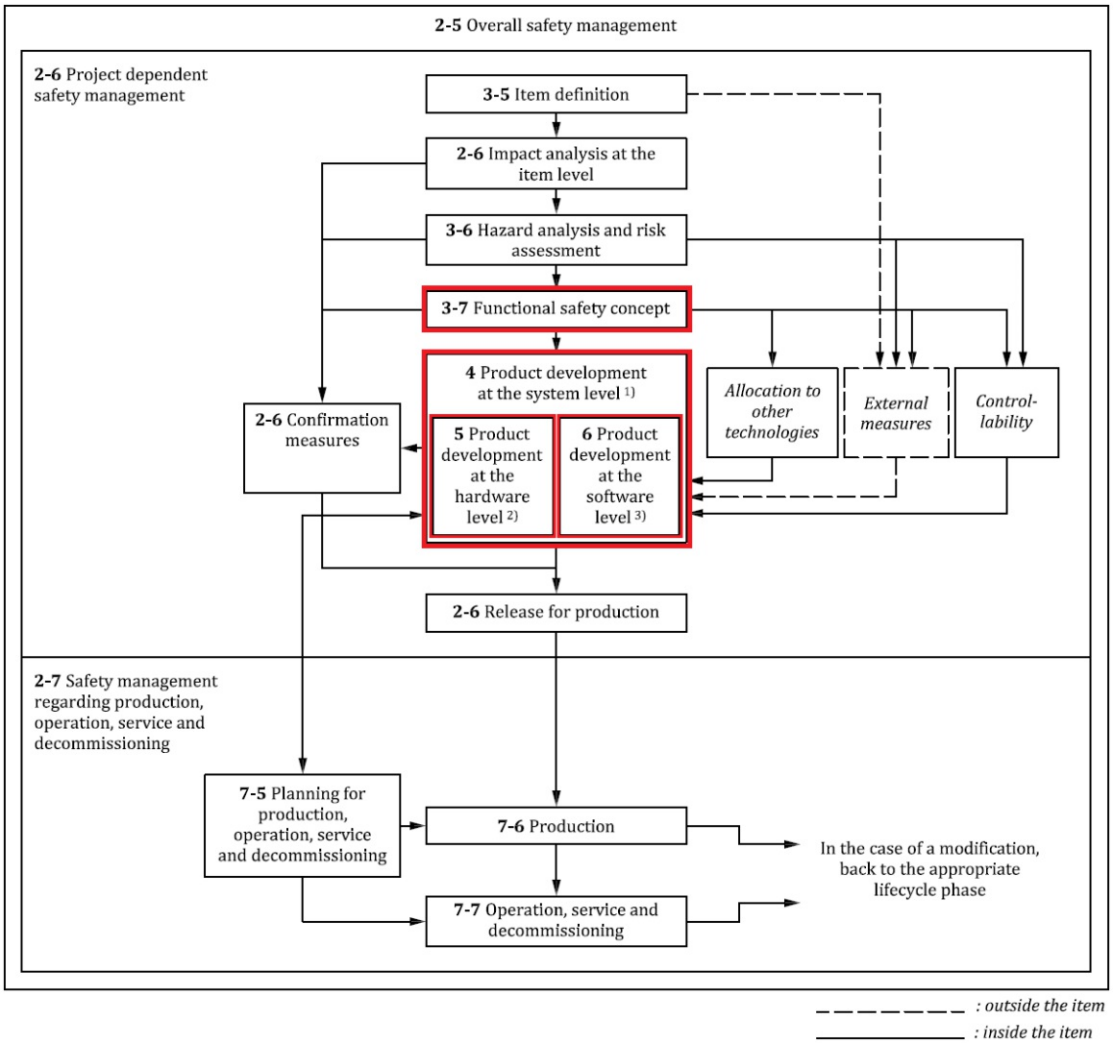

Figure 4.2: ISO 26262 management activities in relation to the ISO 26262 safety life cycle (adapted from [ISOf, p. 5]).

item and estimates the probability of exposure, the controllability and the severity of the hazardous events with regard to the item. The outcome of this activity are safety goals with assigned ASILs (see Figure 4.3) [ISOg, p. 5-11]. The definition of safety goals and the determination of ASILs are crucial for functional safety management.

(d) Functional safety concept: After the safety goals have been defined, the functional safety concept can be developed by deriving functional safety requirements from the safety goals as illustrated in Figure 4.3. Furthermore, the functional safety requirements are allocated to the system architectural design. The advantage of this early assignment to the architectural design is that the use of preliminary architec-

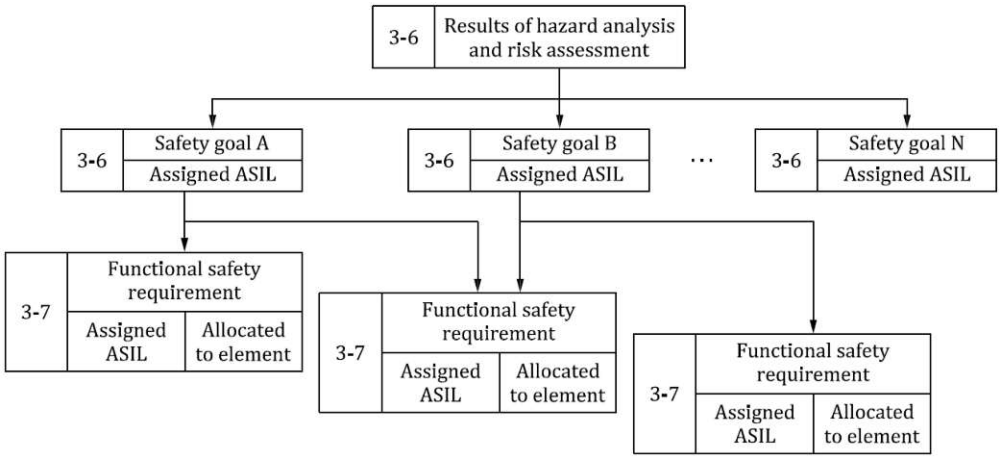

Figure 4.3: ISO 26262 hierarchy of safety goals and functional safety requirements [ISOg, p. 13].

tural assumptions provides a means to handle immature architectural information in early development phases [ISOg, p. 12f]. **As soon as the functional safety concept information is available, ASIL decomposition can be performed for the first time** [ISOm, p. 5].

(e) Product development: In this activity, the item is developed at the system level. It usually starts with the definition of a technical safety concept. After the technical safety concept is determined, the hardware and software are developed [ISOf, p. 2f]. Both the hardware and software development processes are based on the V-model concept with the specification of the requirements and the architectural design and implementation on the left side, and the integration and the verification on the right side. After the hardware and software development have been finished individually, both are integrated at the vehicle level for testing and safety validation to provide evidence of functional safety with respect to the safety goals [ISOh, p. 4]. **ASIL decomposition can be applied for requirements at the system level, at the hardware level, and at the software level**. The requirements are also detailed in a hierarchical structure based on the individual activities and therefore the decomposition differs from level to level [ISOm, p. 5]. The structure and dependencies of safety requirements in the context of ISO 26262 are illustrated in Figure 4.4.

(f) Confirmation measures: In the second part of ISO 26262 (see [ISOf]), required confirmation measures are listed to conform to the functional safety of items. This confirmation is justified based on confirmation reviews, functional safety audits, and functional safety assessments. Furthermore, the ISO 26262 specifies four levels of independence, and the requirements are depending on the ASIL level of the measured item. The levels of independence determine by whom the confirmation

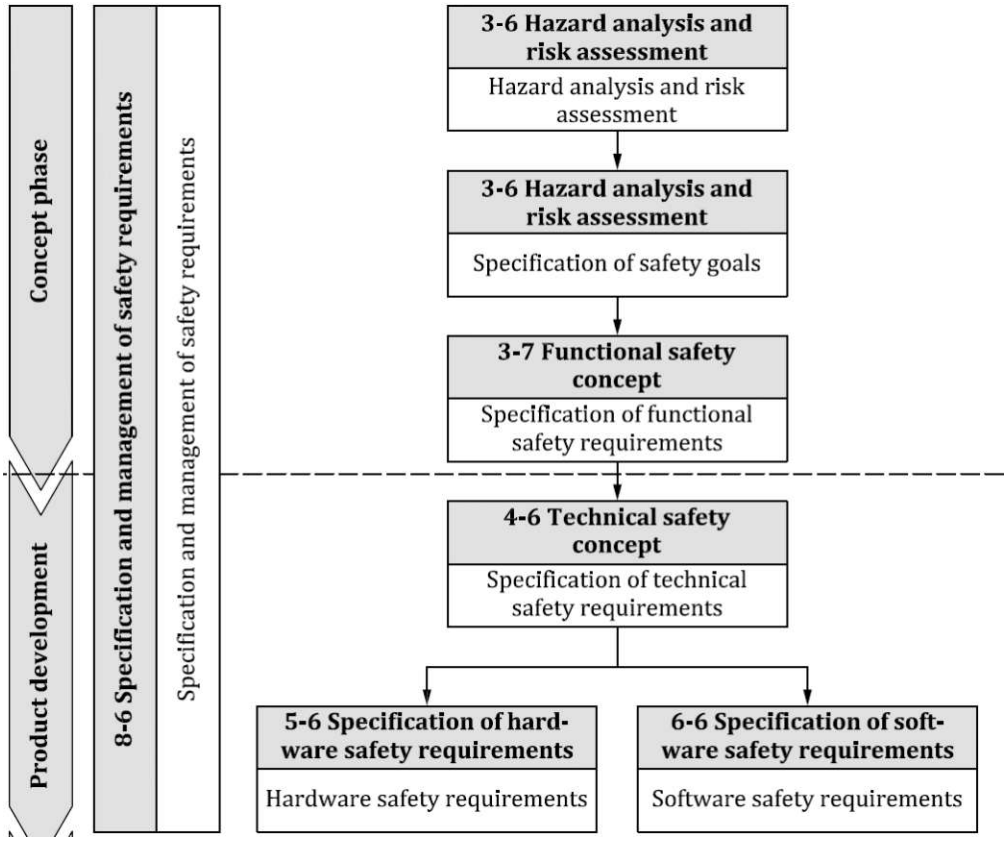

Figure 4.4: ISO 26262 structure and dependencies of safety requirements [ISOm, p. 10].

measure shall be performed. For instance, independence level 3 requires a person who is independent, regarding management, resources and release authority, from the department responsible for the creation of the considered work product for the performed confirmation measure [ISOf, p. 20-26].

(g) Release for production: The confirmation measures as described above confirm the functional safety of items. These confirmation measures are required for the development of safety cases. Safety cases furthermore provide a crucial argument for the achievement of functional safety. The release for production of the item shall only be approved if there is sufficient evidence for confidence in the achievement of functional safety and complete documentation is available. Thus, confirmation measure reports and safety cases are a must for the release for production.

(h) (Planning for) Production, operation, service, and decommissioning: The planning for production, operation, service, and decommissioning starts during the product development and takes place at the same time as the product development. The production phase ensures that functional safety is achieved during the production phase. It mainly addresses original equipment manufacturer (OEM) suppliers

because the production of items and elements is rarely done by the vehicle manufacturer itself. Operation, service, and decommissioning comprise field monitoring for safety-related incidents, compliance with the service plans, and change management according to ISO 26262 requirements [ISOk]. However, this phase is out of scope for this work.

In summary, as already noted, the ASIL decomposition method can be applied in four different phases. Namely, in the functional safety concept phase that specifies functional safety requirements, in the system design phase which specifies technical safety requirements, in the hardware design phase that specifies hardware safety requirements, or at the software design phase that specifies software safety requirements. It is important to note that safety requirements can also be nested. These four phases define the addressed elements for the SysML extension developed in section 4.3.

Now that the basic concepts of ASIL tailoring and decomposition have been explained, requirements that have to be fulfilled can be examined more closely. In table 4.1, the requirements that must be fulfilled to apply ISO 26262 compliant ASIL decomposition are listed. It should be noted that it is not reasonable or possible to carry out all requirements in the sense of model-based engineering. Thus, we have not listed all requirements as listed in [ISOm].[3]

Table 4.1: ASIL decomposition requirements according to [ISOm, p. 5ff].

RQ 5.4.2	ASIL decomposition shall be performed by considering each initial safety requirement individually.
RQ 5.4.3	The initial safety requirement shall be decomposed to redundant safety requirements that shall be implemented by sufficiently independent elements. Note: This does not work for dependent failures.
RQ 5.4.4	Each decomposed safety requirement shall comply with the initial safety requirement by itself.
RQ 5.4.7	If ASIL decomposition of an initial safety requirement results in the allocation of decomposed requirements to the intended functionality and an associated safety mechanism, a safety requirement shall be allocated to the intended functionality and implemented applying the corresponding decomposed ASIL.
RQ 5.4.8a RQ 5.4.9	When applying ASIL decomposition to a safety requirement, a decomposition schema must be chosen in accordance with the ASIL before the decomposition (see 4.16).
RQ 5.4.8b	When applying ASIL decomposition to a safety requirement, an ASIL decomposition may be applied more than once.

[3]The requirements 5.4.1, 5.4.5, 5.4.6, 5.4.10, 5.4.11, and 5.4.12 are excluded from this work since systems engineering using MBSE is not feasible for these requirements.

RQ 5.4.8c	When applying ASIL decomposition to a safety requirement, each decomposed ASIL shall be marked by putting the ASIL of the safety goal in parentheses.

This table also defines some of our validation criteria for the practical application in section 4.3 and most of the criteria for the performed descriptive evaluation in section 4.4.

4.2.2 Design-Science Research

In the information system (IS) research discipline, there are two essential and characterizing paradigms. Namely, the behavioral science paradigm and the design-science paradigm [HRM+04]. Hevner et al. described both more precisely as follows:

> *The behavioral science paradigm seeks to develop and verify theories that explain or predict human or organizational behavior. The design-science paradigm seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts* [HRM+04, p. 75].

In [HRM+04], the authors presented a design-science framework for research in IS to "describe the performance of design-science research (...) via a concise conceptual framework and clear guidelines for understanding, executing, and evaluating the research". This framework supporting design-science research is applied for the development of the DSL in this work. Nevertheless, this paradigm is still not uncritical. Alturki et al. [AGB13] stated that design-science research lacks a comprehensive and detailed methodology. The authors of [GCSTT12] criticized that there is a lack of guidance on the design of the artifact to be created. Furthermore, Bisandu [Bis16] presented a literature review on the design-science research in IS in 2016 and noted the advantage that the method solves many crucial problems through artifact development. However, there is no further discussion provided in this thesis about the advantages and disadvantages of design-science research. The focus in this section is on the framework presented by Hevner et al. [HRM+04] and its adaption illustrated in Figure 4.6 for this work's research goal. The conducted research is furthermore assisted by the following seven guidelines for effective design-science research as stated in [HRM+04, p. 83]:

1. *Design as an Artifact: Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.*
2. *Problem Relevance: The objective of design-science research is to develop technology-based solutions for important and relevant business problems.*
3. *Design Evaluation: The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.*

4. *Research Contributions: Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.*
5. *Research Rigor: Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.*
6. *Design as a Search Process: The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.*
7. *Communication of Research: Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.*

These seven guidelines are considered during the DSL's systematic development. However, the focus is on the Information Systems Research Framework that combines behavioral-science and design-science paradigms as suitable for this research (see Figure 4.5). The advantages are the clarity and easier description of the conducted research.

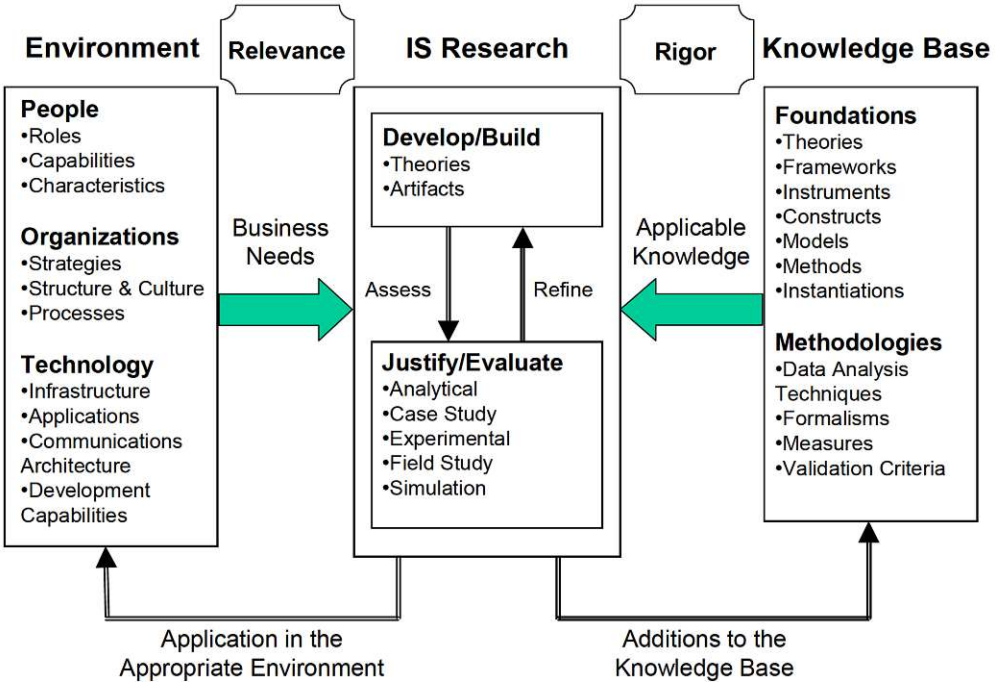

Figure 4.5: Information Systems Research Framework [HRM+04, p. 80].

The Information Systems Research Framework has been adapted for this work and its instance is illustrated in Figure 4.6. In the following sub-subsections 4.2.2.1-4.2.2.3, all parts of the adapted framework are described in order to build an artifact with an appropriate and fully evaluated utility.

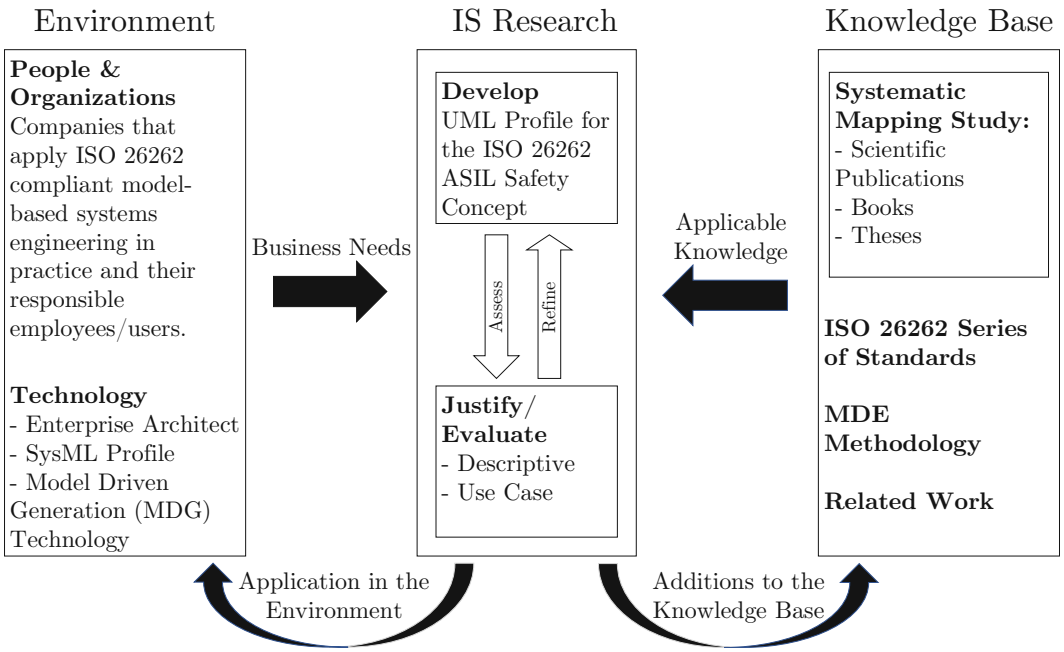

Figure 4.6: Adapted Information Systems Research Framework for the *ASIL SysML Profile* DSL development (instantiation of [HRM+04, p. 80]).

4.2.2.1 Environment

The environment defines the problem space and is composed of organizations and their people. Additionally, existing technologies of the organization are used or new technologies are planned and part of the environment [HRM+04, p. 79].
For this research, we address all companies that apply ISO 26262 compliant model-based systems engineering already in practice. Since ASIL decomposition modeling in SysML represents only a small part of an ISO 26262 systems engineering process, it cannot be used as a stand-alone modeling tool in a meaningful way. Furthermore, the created artifact must be described in a way that key users who already apply ISO 26262 compliant model-based systems engineering must be able to use the created DSL in practice.[4]

Apart from the people's and organizations' space, the environment also includes technical aspects. Our development approach uses the Sparx Systems Model Driven Generation (MDG) technology to create the SysML extension. Further details about the MDG technology are provided in section 4.3. The two domains people and organizations, and

[4]Normally an evaluation would have to be carried out to prove the truth of this statement (e.g. case study). Such an evaluation is not part of this thesis and is recommended for future work.

technologies both define our business need that frames our research activities conducted in the middle part of our framework.

4.2.2.2 Knowledge Base

The knowledge base is located in the right part of the framework and provides, as Hevner et al. have noted, "the raw materials from and through which IS research is accomplished" [HRM+04, p. 80]. The Systematic Mapping Study conducted in chapter 3 and other related work provide information about existing theories, frameworks, and research about model-based safety assurance approaches in the context of ISO 26262. These two sources are essential for identifying problem gaps in order to achieve a useful addition to the knowledge base. The addition to the knowledge base is derived from the created artifact[5].

Additionally, parts 1 to 10 of the ISO 26262 series of standards are part of the knowledge base (see [ISOd, ISOf, ISOg, ISOh, ISOi, ISOj, ISOk, ISOl, ISOm, ISOb]). Parts 11 (see [ISOc]), entitled *Guidelines on application of ISO 26262 to semiconductors*, and 12 (see [ISOe]), entitled *Adaptation of ISO 26262 for motorcycles*, are not relevant for the research conducted in this work. Thus, these two parts of the ISO 26262 series of standards are excluded from the knowledge base.

The third part of the knowledge base is the model-driven engineering (MDE) methodology. It consists of the Unified Modeling Language (UML) specification in version 2.5.1 (see [Gro17b]). Furthermore, the SysML specification in version 1.6 (see [Gro19a]) is part of the knowledge base. However, the UML and SysML specifications provide no guidance for the development of a domain-specific artifact. For this reason, the development process follows Strembeck's & Zdun's [SZ09] approach for the systematic development of domain-specific languages. It defines the main and sub-activities when engineering a DSL. The executed MDE approach is described in detail in section 4.3 and evaluated in section 4.4.

The publications listed as related work in the knowledge base are discussed in subsection 4.1.1. Since some work has already been carried out successfully in the field of research to assure partly ISO 26262 compliance with UML profiles, these publications are considered for our design-science research. The knowledge base serves as applicable knowledge in the develop/build and justify/evaluate cycle within the IS research loop.

4.2.2.3 Information System (IS) Research

IS research is the most crucial part of the applied research framework and is conducted in two complementary phases. Hevner et al. [HRM+04, p. 80f] described one phase as "behavioral science addresses research through the development and justification of theories that explain or predict phenomena related to the identified business need (...)

[5]The created artifact in this work is a developed and fully evaluated DSL.

the goal of behavioral-science research is truth", and the other phase as "design science addresses research through the building and evaluation of artifacts designed to meet the identified business need (...) the goal of design-science research is utility". The interdependence between the development phase and the justify respectively evaluate phase is illustrated in the middle section of Figure 4.6.

The development phase creates the *ASIL SysML Profile* DSL to define an appropriate means to represent the ASIL decomposition concept in the context of ISO 26262. The justify/evaluate phase is responsible for the proper design evaluation. The descriptive evaluation performs an informed argument evaluation method that mainly uses ISO 26262-9 [ISOm], in addition to other sources from the knowledge base, to build a convincing argument for the SysML extension's utility. Furthermore, a use case shows a detailed real-life scenario using the developed DSL to demonstrate its utility [HRM+04, p. 86]. In summary, these two phases alternate several times in the context of assess and refine loops during the design-science research conducted in this work. The assess and refine loops are especially influenced by Strembeck's & Zdun's [SZ09] approach. All four main activities must be completed besides other requirements until the development is finished. In the following section 4.3, a detailed discussion is provided on how the SysML is extended using MDG to build a SysML profile.

4.3 Practical Approach and Technical Solution

This section describes the technical aspects of the SysML and the creation and evaluation processes of the *ASIL SysML Profile* DSL. This chapter is structured as follows. Firstly, the extension of the SysML using a SysML profile is described. Secondly, each of the four main activities for the systematic development of a DSL presented from Strembeck & Zdun [SZ09] is performed individually and described step by step. Third, a SysML use case is presented to show the application of ASIL tailoring and decomposition support by the *ASIL SysML Profile*. The use case introduces a model-based systems engineering example of an ISO 26262 compliant car dashboard. In the end, the final evaluation is described. The DSL's final evaluation is performed as a descriptive evaluation based on twenty-two evaluation criteria.

4.3.1 Introduction

Before the development is carried out systematically, this subsection briefly introduces the SysML and how it is extended. The SysML reuses a subset of UML 2 and provides additional extensions needed to address requirements in the UML for Systems Engineering [Gro19a]. It is defined as a profile of UML 2 and reuses most of its modeling constructs. Furthermore, new modeling constructs are defined for SysML that have no counterparts in UML or replace UML constructs [Sys]. It is possible to extend the SysML by defining an extending SysML profile to overcome some of its limitations [AZ13]. With regard to the problem statement of this thesis, the SysML is extended to facilitate the application

of ASIL decomposition using model-based systems engineering. The relationship between the UML and SysML languages is illustrated in Figure 4.7. The left part of the Venn diagram marked in red highlights all SysML modeling constructs, some of which we extend.

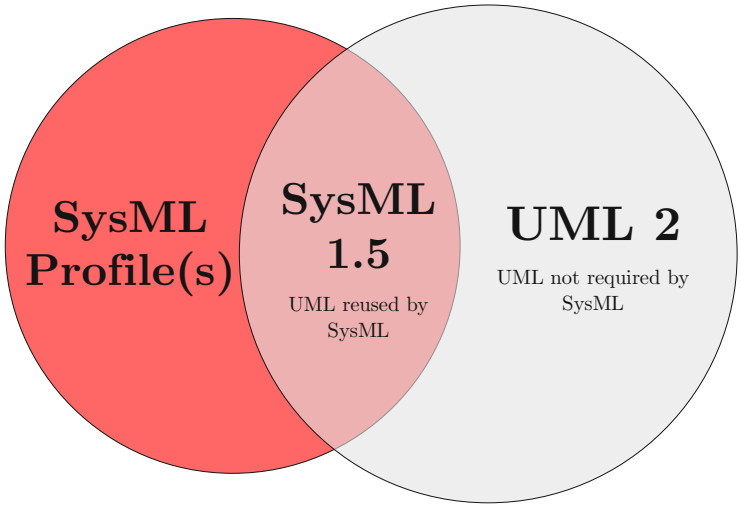

Figure 4.7: SysML/UML interrelationship (adapted from [Gro17a, p. 9]).

The implementation of the SysML profile uses the MDG technology by Sparx Systems (see [Spaa]). Since the development of the SysML profile has the same requirements as the creation of a new DSL, it is not a trivial task. Due to this fact, the development process follows Strembeck's & Zdun's [SZ09] proposed systematic DSL development method. The four main activities recommended by the authors are carried out in the following subsection 4.3.2 to create the *ASIL SysML Profile* systematically.

4.3.2 DSL Creation Activities

In this subsection, we describe the DSL engineering process to define an appropriate means of representing the ISO 26262 ASIL tailoring and decomposition concept in a SysML profile. In the first step of this engineering process (see 4.3.2.1), both UML language elements and SysML language elements are extended to define a core language model. Furthermore, specific validation constraints are defined in this step. The DSL's behavior is defined in the second step (see 4.3.2.2) and the concrete syntax of the DSL in the third step (see 4.3.2.3). In the fourth and last step (see 4.3.2.4), the Enterprise Architect[6] application is used to perform a target platform for the DSL.

[6] https://www.sparxsystems.eu/ea15/

4.3.2.1 Definition of the DSL's core language model (Step 1)

As a first step, the DSL's core language model is defined. The SysML diagram taxonomy as illustrated in Figure 4.8 comprises several diagrams. For the *ASIL SysML Profile* DSL, it is only necessary to extend the *Requirement Diagram*, the *Block Definition Diagram*, and the *Internal Block Diagram*.

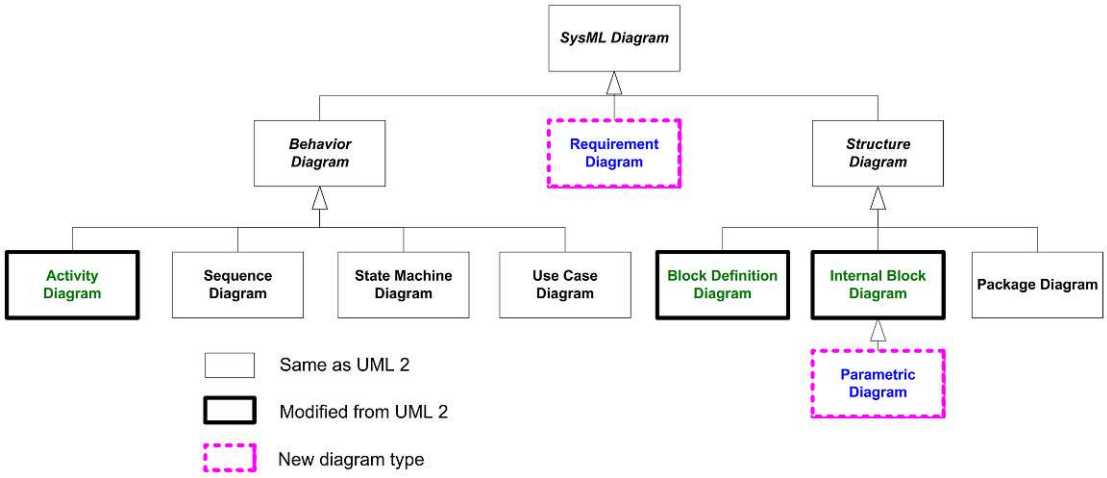

Figure 4.8: SysML version 1.5 diagram taxonomy [Gro17a, p. 194].

These different diagrams are not explained in detail but more information can be found in [Gro19a]. In order to model the concepts of ASIL tailoring and ASIL decomposition, it is necessary to extend the model element *requirement* from the *Requirement Diagram*, the model element *block* from the *Block Definition Diagram*, and the model element *Part* from the *Internal Block Diagram*. Furthermore, new dependencies are essential to link the newly defined elements. Figure 4.9 depicts the core language model for the ASIL tailoring and decomposition domain as a UML class model based on extensions and generalizations. All new stereotypes are described subsequently.

The major goal has been to extend SysML modeling elements that already exist rather than defining new ones. The modeling elements are now described in the same order as they are used in the workflow. Furthermore, the references to the specification of the SysML elements are provided. The attribute *ASIL* is represented as an enumeration that covers all possible safety levels including decomposition ones to avoid mistakes. The *ASIL* is included in all new safety-related elements beside the two dependencies *assignedTo* and *decomposedFrom*.

A *safetyBlock* specializes a SysML *block*[7] by the attributes *ASIL* and *decomposed*. A *block* contains the SysML internal block elements *property*. Both the *property* and the

[7]Specified in [Gro19a, p. 32]

safetyProperty extend the metaclass *Part*[8]. This is necessary to set a *safetyProperty* as property element of a *safetyBlock*. Furthermore, the modeling elements *functionalSafetyProperty*, *softwareSafetyProperty*, and *hardwareSafetyProperty* specialize the *safetyProperty* and can be used to model internal elements of a *safetyBlock*. Technical safety properties are not explicitly modeled as they can always be divided into software and hardware concerns. Thus, the *abstractTechnicalSafetyProperty* is abstract to avoid mistakes due to the wrong application of this element. If ASIL decomposition is applied, the *functionalSafetyPropery*, *softwareSafetyProprety* or *hardwareSafetyProperty* is split up into two redundant elements, and the dependency *decomposedFrom* labels the decomposition hierarchy in the context of safety properties. The stereotype *decomposedFrom* is a standard dependency extending the metaclass *Dependency*[9] to add meaning to the link without adding any new features.

The stereotype *safetyRequirement* specializes the SysML stereotype *requirement*[10]. No more attributes than the *ASIL* are required for this stereotype as a simplified approach is followed. Since a *safetyRequirement* is created based on the *safetyGoal* that is the outcome of the *Hazard Analysis and Risk Assessment*, these two do not differ except for three attributes. A *safetyGoal* requires the definition of the extent of *controllability*, *exposure*, and *severity*. Same as for the attribute *ASIL*, three additional enumerations are defined to avoid mistakes. A *safetyGoal* is assigned to a *safetyProperty* using the newly defined stereotype *assignedTo*. The stereotype *assignedTo* is a standard dependency extending the metaclass *Dependency* without adding any new features. In order to derive a *safetyRequirement* from a *safetyGoal* and to derive safety requirements from the most generic stereotype down to the most specialized stereotypes, the existing SysML Dependency *deriveRqt*[11] is used. For technical aspects, specific software and hardware safety requirements should always be defined rather than more general technical safety requirements. Thus, the stereotype *abstractTechnicalSafetyRequirement* is abstract to only allow the modeling of its two specializing stereotypes *hardwareSafetyRequirement* and *softwareSafetyRequirement*.

The nested hierarchy for safety properties and safety requirements is required because one level of the hierarchy must not be skipped when modeling. Furthermore, multiplicities of the modeling elements have not been described because they do not differ from the generalized SysML model elements. The syntactical rules for SysML modeling should not change in order to enable a quick familiarization for existing key users.

The UML class diagram as depicted in Figure 4.9 covers all *ASIL SysML Profile* DSL model elements.

[8]Specified in [Gro19a, p. 39]
[9]Specified in [Gro19a, p. 36]
[10]Specified in [Gro19a, p. 191]
[11]Specified in [Gro19a, p. 190]

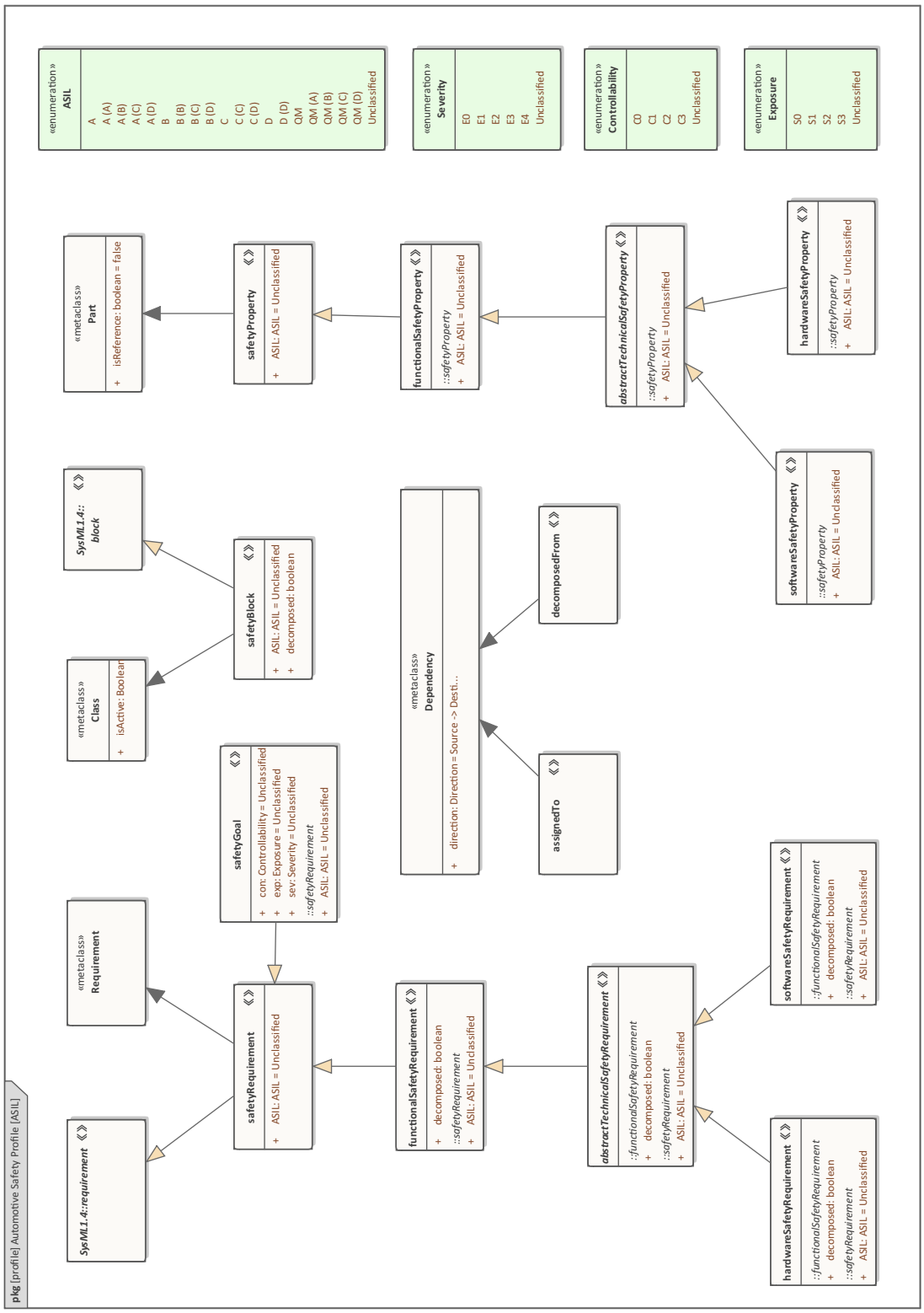

Figure 4.9: *ASIL SysML Profile* DSL elements.

In addition, specific constraints are necessary to ensure consistent and compliant modeling in the context of ISO 26262[12]. The constraints are listed and further described using free text notation in Table 4.2. These constraints are transformed into OCL constraints in the fourth step of the systematic development process described in section 4.3.2.4.

Table 4.2: Constraints defined for the DSL core language model.

SRQ_1[13]	A *safetyRequirement* or an element that specializes the entity *safetyRequirement* must not have the attribute *ASIL* with the assigned value "Unclassified".
SRQ_2	A *safetyRequirement* must be derived (dependency: *deriveRqt*) from a *safetyGoal*.
SRQ_3	A *functionalSafetyRequirement* must be derived (dependency: *deriveRqt*) from a *safetyRequirement*.
SRQ_4	An instance of a *technicalSafetyRequirement* is not allowed because it is an abstract modeling element.
SRQ_5	A *hardwareSafetyRequirement* or a *softwareSafetyRequirement* must be derived (dependency: *deriveRqt*) from a *functionalSafetyRequirement*.
SRQ_6	A decomposed *functionalSafetyRequirement* or a decomposed element that specializes the entity *functionalSafetyRequirement* must be decomposed (dependency: *decomposedFrom*) from a parent element that has the same stereotype.
SRQ_7	A decomposed *functionalSafetyRequirement* or a decomposed element that specializes the entity *functionalSafetyRequirement* must mark the attribute *ASIL* by putting the *ASIL* of the parent element in parentheses.
SRQ_8	A *functionalSafetyRequirement* or an element that specializes the entity *functionalSafetyRequirement* that is involved in an *ASIL* decomposition (dependency: *decomposedFrom*) must agree to the *ASIL* decomposition schema as stated in [ISOm, p. 6f].
SRQ_9	A *functionalSafetyRequirement* or an element that specializes the entity *functionalSafetyRequirement* must have the attribute *decomposed* with the assigned value "true" if it is decomposed into two or more requirement modeling elements.
SRQ_10	A decomposed *functionalSafetyRequirement* or a decomposed element that specializes the entity *functionalSafetyRequirement* must have at least two child elements *decomposed* (dependency: *decomposedFrom*) that have the same stereotype as the parent has.
SRQ_11	A derived *safetyRequirement* or a derived element specializing the entity safetyRequirement (dependency: *deriveRqt*) must not have a different *ASIL* than the element it is derived from.[14]

[12] See sub-subsection 4.2.2.2 for more information about the relevant parts of ISO 26262.
[13] SRQ: Conditions for the modeling element *safetyRequirement*.
[14] This is only allowed by applying *ASIL* decomposition.

SRQ_12	A *safetyRequirement* or an element that specializes the entity *safetyRequirement* must have the same (or lower) ASIL as its satisfying (dependency: *satisfy*) *safetyProperty* or specialization of the entity *safetyProperty*.
SG_1[15]	A *safetyGoal* can only be assigned to (dependency: *assignedTo*) a *safetyBlock*, a *safetyProperty*, or an element that specializes the entity *safetyProperty*.
SG_2	A *safetyGoal* must have the same ASIL as the *safetyBlock*, *safetyProperty* or element that specializes the entity *safetyProperty* it is assigned to (dependency: *assignedTo*).
SG_3	A *safetyGoal* must not have the attribute *ASIL* with the assigned value "Unclassified".
SG_4	A *safetyGoal* must not have the attribute *con* with the assigned value "Unclassified".
SG_5	A *safetyGoal* must not have the attribute *exp* with the assigned value "Unclassified".
SG_6	A *safetyGoal* must not have the attribute *sev* with the assigned value "Unclassified".
SP_1[16]	A *safetyProperty* or an element that specializes the entity *safetyProperty* must not have the attribute *ASIL* with the assigned value "Unclassified".
SP_2	A decomposed *safetyProperty* or a decomposed element that specializes the entity *safetyProperty* must mark the attribute *ASIL* by putting the *ASIL* of the parent element in parentheses.
SP_3	A decomposed *safetyProperty* or a decomposed element that specializes the entity *safetyProperty* must satisfy a decomposed *functionalSafetyRequirement* or a decomposed element that specializes the entity *functionalSafetyRequirement*.
SP_4	A *softwareSafetyProperty* must and must only satisfy a *softwareSafetyRequirement*.
SP_5	A *hardwareSafetyProperty* must and must only satisfy a *hardwareSafetyRequirement*.
SP_6	A *functionalSafetyProperty* must and must only satisfy a *functionalSafetyRequirement*.
SP_7	If a *safetyProperty* or an element that specializes the entity *safetyProperty* is made of sub-elements from the same stereotype, all elements must be treated in accordance with the highest occurring ASIL.
SP_8	A decomposed *functionalSafetyProperty* or a decomposed element that specializes the entity *functionalSafetyProperty* must have at least two child elements decomposed (dependency: decomposedFrom) that have the same stereotype as the parent.

[15]SG: Conditions for the modeling element *safetyGoal*.
[16]SP: Conditions for the modeling element *safetyProperty*.

Now the DSL's core language elements are defined including all modeling elements and validation constraints. In the next sub-subsection 4.3.2.2, the behavior of the DSL is described as a second step of the systematic development.

4.3.2.2 Definition of the DSL language element's behavior (Step 2)

In this second step of the systematic development of the *ASIL SysML Profile*, the behavior concerning the different modeling elements is described. The behavior can best be described by a sequence of activities. The sequence of activities when applying ASIL decomposition with the help of the developed DSL is illustrated by creating a UML Activity Diagram [Gro17b, p. 698ff]. The advantage of the description in a UML Activity Diagram via the description in free text is the better overview of the entire workflow and the visualization of conditions and parallel processing during the execution of activities. Furthermore, swimlanes can be used to group activities executed by specific modeling elements. Figure 4.10 illustrates the workflow and thus the behavior of each modeling element. Actions and activities that are executed several times rather than only once are marked with the condition *[for each x]*. Modeling each loop would greatly reduce the clarity of the diagram. The concern of each new stereotype extending a SysML stereotype or a SysML metaclass that has been defined in the first step (see sub-subsection 4.3.2.1) is modeled in a separate swimlane. More precisely, the activity diagram distinguishes between *safetyBlock*, *safetyProperty*, *safetyGoal*, and *safetyRequirement* concerns.

The first two actions are the item definition at the block level followed by the item definition at the property level. The outcomes are a *safetyBlock* and its *safetyProperty* elements. The *safetyProperty* swimlane comprises the element *safetyProperty* itself and all stereotypes specifying a *safetyProperty*, more precisely, the creation of *functionalSafetyProperty*, *softwareSafetyProperty*, and *hardwareSafetyProperty* elements. All these elements together serve as the input required for the activity *Hazard Analysis and Risk Assessment* that defines a *safetyGoal* that determines the *ASIL* for the first time. The *ASIL* of a *safetyGoal* is based on the classification results for *severity*, *controllability*, and *exposure* for each *safetyProperty*. Afterward, the *safetyRequirement* is broken down from the corresponding *safetyGoal*. A *safetyRequirement* cannot be decomposed. This is only possible from the functional level on. Up to this point in the workflow, the following elements were defined: *safetyBlock*, *safetyProperties*, *safetyGoals* and their *ASILs*, and *safetyRequirements* and their *ASILs*. It is now possible to derive more specialized requirements from generic requirements and apply optional ASIL decomposition at different safety hierarchy levels.

Next in the workflow, *FunctionalSafetyRequirement* elements are derived from *safetyRequirement* elements. If desired, ASIL decomposition can now be applied at the functional safety level, and two or more decomposed *functionalSafetyRequirement* elements can be created. In the next step, *technicalSafetyRequirement* elements are derived implicitly, but not explicitly modeled and decomposed because they are abstract. This is important because based on this action the distinction between software and hardware properties is

determined. Even if *hardwareSafetyProperty* and *softwareSafetyProperty* elements do not differ in our approach, the distinction is important for further model-based development methods such as code generation. The derivation and decomposition activities for both property types is done in the same way. Thus, the activities are only described for elements concerning hardware in Figure 4.10. The *hardwareSafetyRequirement* elements are directly derived from *functionalSafetyRequirement* elements in the model. As already mentioned, the abstract technical safety level is not represented in the model. The created *hardwareSafetyRequirement* is then linked to a *hardwareSafetyProperty*. After these two elements have been linked with each other by a *satisfy* dependency, optional ASIL decomposition at the hardware level can be applied to lower the ASIL under certain circumstances[17]. If ASIL decomposition is applied, the *hardwareSafetyRequirement* is split up into two or more redundant *hardwareSafetyRequirement* elements. The corresponding *hardwareSafetyProperty* is also split up into redundant decomposed elements from the same stereotype. After the decomposition for requirements and properties has been finished, the newly created decomposed elements must be linked with each other with a new *satisfy* dependency. Furthermore, the decomposition hierarchy of requirements and properties is characterized by the *decomposedFrom* dependency. *SoftwareSafetyProperty* and *softwareSafetyRequirement* elements can be derived and decomposed in the same way. The workflow is finished as soon as the modeling at the hardware and software safety level is completed.

The created UML Activity Diagram describes the behavior of the DSL in the form of a workflow based on activities and actions. Each activity and action have now been described in detail. In the third and next step (see sub-subsection 4.3.2.3), the concrete syntax of the developed DSL is defined.

[17] See subsection 4.1.2 and table 4.2 for more information.

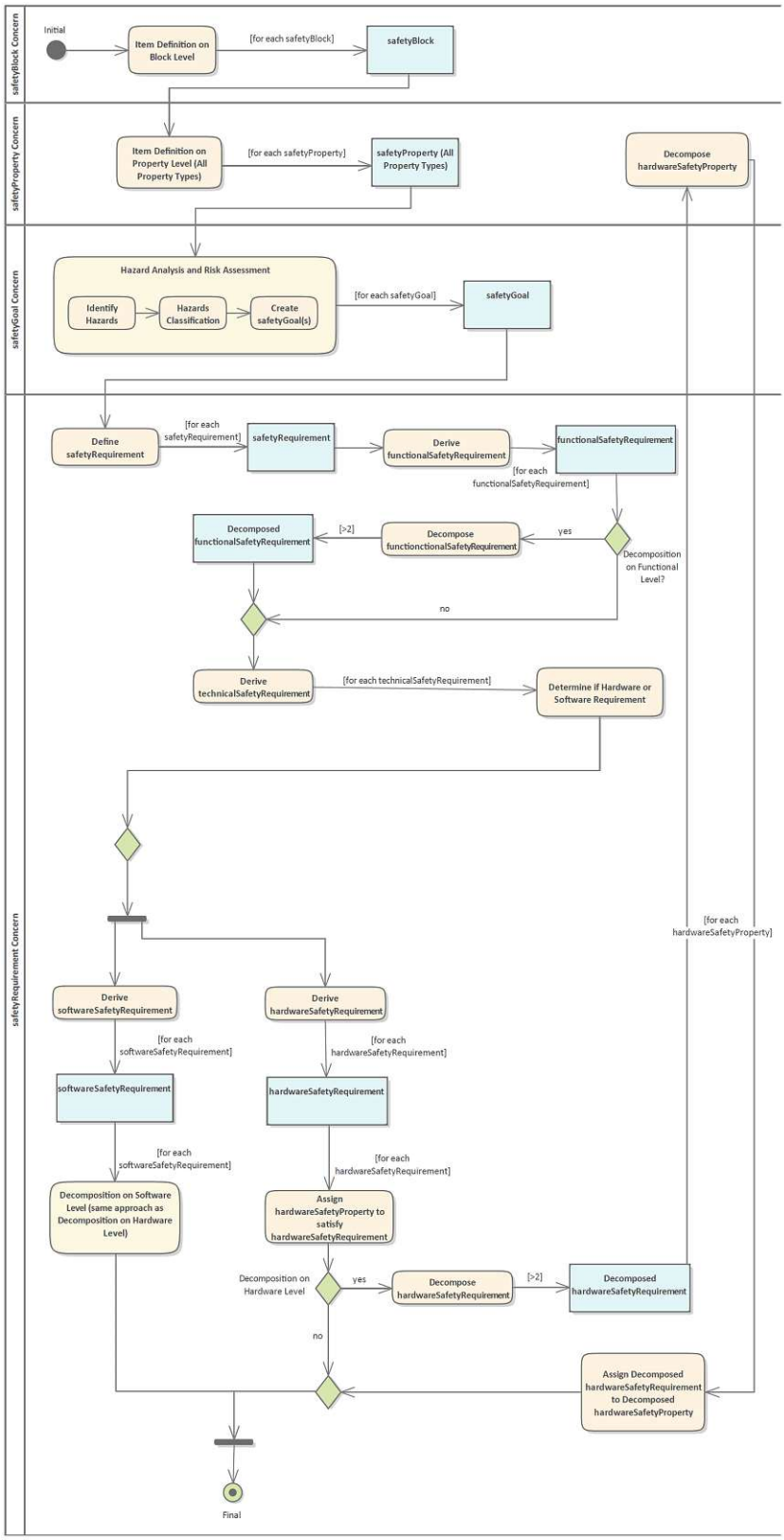

Figure 4.10: UML Activity Diagram illustrating the control flow of the created ASIL decomposition process.

4.3.2.3 Definition of the DSL's concrete syntax (Step 3)

In this third step of the systematic development, the concrete syntax for the *ASIL SysML Profile* DSL is defined. In the illustrative example by Strembeck & Zdun [SZ09, p. 1280f], an XML-based concrete syntax has been developed. A textual syntax is one possible approach to define a DSL's concrete syntax. For the DSL developed in this work, the SysML language has been extended. Due to the fact that SysML is a graphical modeling language, the definition of a textual syntax would be pointless. Thus, a graphical concrete syntax has to be defined to specify a visual notation.[18] The graphical syntax is defined below.

The developed graphical concrete syntax only covers modeling elements that are relevant for the ASIL decomposition SysML profile. Since the SysML stereotypes *requirement* and *block* defined in [Gro19a] have been extended, many artifacts of these stereotypes are inherited to the extending safety-specific modeling elements. For this reason, inherited artifacts such as attributes or operations that are not necessary for this work are not listed in the graphical concrete syntax. Furthermore, safety-specific property modeling elements have been defined as new stereotypes that extend the metaclass *Part*. *Parts* can be used to express composite structures to accomplish a particular domain-specific purpose [Spab]. The defined graphical concrete syntax is visualized in Table 4.3. The definition has been inspired by OMG's SysML concrete syntax specifications in [Gro19a].

Table 4.3: *ASIL SysML Profile* graphical concrete syntax.

«safetyGoal»	*«safetyGoal»* **Safety Goal Name** *Derived* «safetyRequirement»Master SftReqt Name *AssignedTo* «softwareSafetyProperty»SoftSftProp Name «hardwareSafetyProperty»HardSftProp Name «functionalSafetyProperty»FuncSftProp Name Controllability = Unclassified Exposure = Unclassified Severity = Unclassified ASIL = Unclassified

[18] More information about textual and graphical syntaxes can be found in [BCW17].

«safetyRequirement»	*«safetyRequirement»* **Safety Requirement Name** *Derived* «functionalSafetyRequirement»Derived FuncSftReqt Name *DerivedFrom* «safetyGoal»Master SftGoal Name Id = ““ Text = ““ ASIL = Unclassified
«functionalSafetyRequirement»	*«functionalSafetyRequirement»* **Functional Safety Requirement Name** *Derived* «softwareSafetyRequirement»Derived SoftSftReqt Name «hardwareSafetyRequirement»Derived HardSftReqt Name *DerivedFrom* «safetyRequirement»Master SftReqt Name Id = ““ Text = ““ ASIL = Unclassified Decomposed = False
«softwareSafetyRequirement»	*«softwareSafetyRequirement»* **Software Safety Requirement Name** *DerivedFrom* «functionalSafetyRequirement»Master FuncSftReqt Name *SatisfiedBy* «softwareSafetyProperty»SoftSftProp Name *Decomposed* «softwareSafetyRequirement»Dec SoftSftReqt Name 1 «softwareSafetyRequirement»Dec SoftSftReqt Name 2 *DecomposedFrom* «softwareSafetyRequirement»Master SoftSftReqt Name Id = ““ Text = ““ ASIL = Unclassified Decomposed = False

«hardwareSafetyRequirement»	*«hardwareSafetyRequirement»* **Hardware Safety Requirement Name** *DerivedFrom* «functionalSafetyRequirement»Master FuncSftReqt Name *SatisfiedBy* «hardwareSafetyProperty»HardSftProp Name *Decomposed* «hardwareSafetyRequirement»Dec HardSftReqt Name 1 «hardwareSafetyRequirement»Dec HardSftReqt Name 2 *DecomposedFrom* «hardwareSafetyRequirement»Master HardSftReqt Name Id = ““ Text = ““ ASIL = Unclassified Decomposed = False
«safetyBlock»	*«safetyBlock»* **Safety Block Name** *Parts* «property»Block1 Name «property»Block2 Name {subsets Block1} «hardwareSafetyProperty»HardSftProp1 Name «hardwareSafetyProperty»HardSftProp2 Name {subsets HardSftProp1} «softwareSafetyProperty»SoftSftProp1 Name «softwareSafetyProperty»SoftSftProp2 Name {subsets SoftSftProp1} *Properties* «property»Property1 Name ASIL = Unclassified Decomposed = False

«safetyProperty»	*«safetyProperty»* **Safety Property Name** *Block* «safetyBlock»Master Block Name *Properties* «property»Property1 Name *Safety Properties* «functionalSafetyProperty»FuncSftProp Name ASIL = Unclassified
«functionalSafetyProperty»	*«functionalSafetyProperty»* **Functional Safety Property Name** *Block* «safetyBlock»Master Block Name *Properties* «property»Property1 Name *Safety Properties* «hardwareSafetyProperty»HardSftProp Name «softwareSafetyProperty»SoftSftProp Name «safetyProperty»Master SafetyProp Name *Decomposed* «functionalSafetyProperty»Dec FuncSftProp Name 1 «functionalSafetyProperty»Dec FuncSftProp Name 2 *DecomposedFrom* «functionalSafetyProperty»Master FuncSftProp Name *Assign* «safetyGoal»Master SafetyGoal Name ASIL = Unclassified

«softwareSafetyProperty»	*«softwareSafetyProperty»* **Software Safety Property Name** *Block* «safetyBlock»Master Block Name *Properties* «property»Property1 Name *Safety Properties* «softwareSafetyProperty»SoftSftProp Name «softwareSafetyProperty»Master SoftSafetyProp Name *Decomposed* «softwareSafetyProperty»Dec SoftSftProp Name 1 «softwareSafetyProperty»Dec SoftSftProp Name 2 *DecomposedFrom* «softwareSafetyProperty»Master SoftSftProp Name *Satisfy* «softwareSafetyRequirement»SoftSftReqt Name *Assign* «safetyGoal»Master SafetyGoal Name ASIL = Unclassified
«decomposedFrom»	*«functionalSafetyProperty»* or *«softwareSafetyProperty»* or *«hardwareSafetyProperty»* or *«functionalSafetyRequirement»* or *«softwareSafetyRequirement»* or *«hardwareSafetyRequirement»* **Decomposed Safety Property/ Requirement Name 1** *«decomposedFrom»* *«functionalSafetyProperty»* or *«softwareSafetyProperty»* or *«hardwareSafetyProperty»* or *«functionalSafetyRequirement»* or *«softwareSafetyRequirement»* or *«hardwareSafetyRequirement»* **Master Safety Property/ Requirement Name** *«functionalSafetyProperty»* or *«softwareSafetyProperty»* or *«hardwareSafetyProperty»* or *«functionalSafetyRequirement»* or *«softwareSafetyRequirement»* or *«hardwareSafetyRequirement»* **Decomposed Safety Property/ Requirement Name 2** **All three entities must have the same stereotypes!**

«hardwareSafetyProperty»	*«hardwareSafetyProperty»* **Hardware Safety Property Name** *Block* «safetyBlock»Master Block Name *Properties* «property»Property1 Name *Safety Properties* «hardwareSafetyProperty»HardSftProp Name «hardwareSafetyProperty»Master HardSafetyProp Name *Decomposed* «hardwareSafetyProperty»Dec HardSftProp Name 1 «hardwareSafetyProperty»Dec HardSftProp Name 2 *DecomposedFrom* «hardwareSafetyProperty»Master HardSftProp Name *Satisfy* «hardwareSafetyRequirement»HardSftReqt Name *Assign* «safetyGoal»Master SafetyGoal Name ASIL = Unclassified
«satisfy»	*«hardwareSafetyProperty»* **Hardware Safety Property Name** —*«satisfy»*→ *«hardwareSafetyRequirement»* **Hardware Safety Requirement Name** *«softwareSafetyProperty»* **Software Safety Property Name** —*«satisfy»*→ *«softwareSafetyRequirement»* **Software Safety Requirement Name**
«assignedTo»	*«safetyGoal»* **Safety Goal Name** —*«assignedTo»*→ *«safetyProperty»* or *«functionalSafetyProperty»* or *«hardwareSafetyProperty»* or *«softwareSafetyProperty»* **Safety Property Name**

The graphical concrete syntax has now been completely defined. This is of great importance because it represents the user interface of the *ASIL SysML Profile* DSL. The definition has been carried out in the same way as the syntax definition of SysML in [Gro19a]. Thus, it is comprehensible and easy to use for the DSL's key users[19] and domain experts who already have experience in SysML systems modeling. In the next and last step in sub-subsection 4.3.2.4, the platform integration with the Enterprise Architect application platform is carried out.

4.3.2.4 Integrating the DSL artifacts with the platform (Step 4)

The last step in this approach of the systematic development of the *ASIL SysML Profile* DSL is the integration of its artifacts with the target Enterprise Architect application platform. Since all steps in the development process have considered the Enterprise Architect's MDG technology by Sparx Systems to extend the SysML language, there is no further need to map the DSL's artifacts to the target platform or define DSL-to-platform transformations for the modeling language elements. The only exception is the transformation of constraints in free text notation defined in the first step (see sub-subsection 4.3.2.1) to OCL constraints. No specific approach has been followed during the constraints transformations because the transformations are self-contained and have no complex connections between each other. All OCL constraints are listed in Appendix 5. At the moment Enterprise Architect only performs syntax checks for OCL constraints. Further work could focus on creating a validation script using the C# build-in model validation.[20] The platform-specific application of the MDG technology in Enterprise Architect is described subsequently.

The SysML 1.5 is already defined as an MDG technology in the Enterprise Architect application. It is not possible for the user himself or herself to extend an existing MDG. Instead it is necessary for the user to create his or her own MDG technology that extends the core MDG technology. In this work, the core MDG technology is SysML 1.5 MDG, and the extending MDG technology is the *ASIL SysML Profile* MDG. Overall, an MDG may include the following resources: UML profiles, code modules, scripts, patterns, images, tagged value types, report templates, linked document templates, and toolbox pages [Sys17]. The developed *ASIL SysML Profile* MDG uses only a profile to extend the SysML. Sparx Systems defines a profile as a "collection of additional stereotypes and tagged values that extend or are applied to elements, attributes, methods and connectors, which together describe some particular modeling problem and facilitate modeling constructs in that domain" [Sys17, p. 17]. Thus, it is important to mention that a SysML profile has been developed rather than a UML profile.

The created modeling elements of the SysML profile extension have already been presented in sub-subsection 4.3.2.1 and illustrated in Figure 4.9. In summary, the profile was

[19] The key users have been defined in subsection 4.2.2.
[20] See [EAV] for more information.

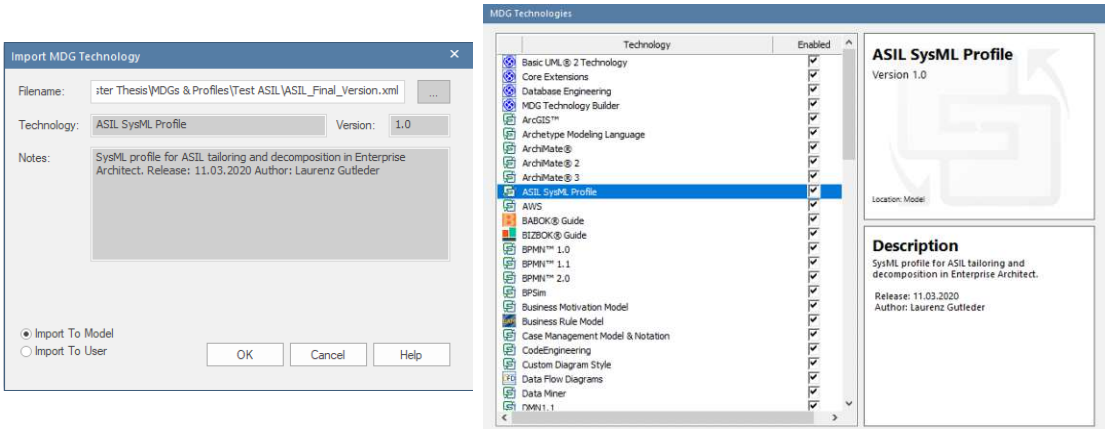

Figure 4.11: Import of the *ASIL SysML Profile* MDG and other provided MDGs.

entirely modeled in Enterprise Architect and consists of additional stereotypes, tagged values, and constraints to define an appropriate means to apply ASIL tailoring and decomposition in the SysML. After the complete SysML profile extension was published, it was incorporated into the new MDG technology, entitled ASIL SysML Profile. The MDG has been stored in an XML file and can be imported and used by any Enterprise Architect application[21]. The XML file is listed in Appendix 5. The import of the *ASIL SysML Profile* MDG using the corresponding XML file and a brief overview of other MDG technologies are shown in Figure 4.11.

All steps according to Strembeck's & Zdun's [SZ09] approach for the systematic development of a DSL have now been successfully carried out. These four steps supported the process from the idea of extending the SysML language to the fully self-developed DSL, entitled *ASIL SysML Profile*. In subsection 4.3.3, an ASIL tailoring and decomposition use case is presented to show the possibilities of the DSL using a real-life example. Following this use case, the DSL's descriptive evaluation and its final results are described in section 4.4.

4.3.3 Use Case

In this subsection, a use case is presented using the *ASIL SysML Profile* DSL that has been systematically developed in the previous subsections 4.3.2.1-4.3.2.4. The goal of the presented use case is to show a simplified example of how the application of the ASIL decomposition concept can be represented and supported by the DSL developed in this work. Normally, individual parts of the system are represented in the SysML in several individual diagrams. For instance, the *Block Definition Diagram* defines a black box representation of the main block, alongside the hierarchy of its composite blocks. Afterward, a white box representation of each block that defines all elements within the system block is modeled in an *Internal Block Diagram*. For our purpose, the individual

[21] Version 15 or higher that includes the SysML 1.5 MDG is required.

model elements of the use case are shown in two diagrams. More precisely, an *Internal Block Diagram* and a *Requirement Diagram* are modeled. This does not influence the syntactic correctness because ASIL decomposition only influences these two diagram types. Nevertheless, this example does not reflect the full extent of what the application of SysML to a real-life system development example would be. The basic idea for this use case is based on [ISOb, p. 57-60] and [PF11]. In Figure 4.10, all necessary activities for ISO 26262 compliant model-based ASIL tailoring and decomposition have already been illustrated and described. Thus, the individual modeling steps are not explained in detail again in this subsection.

The use case covers a part of the overall development of a car dashboard in the context of ISO 26262, more precisely, the part of the dashboard responsible for processing data from the sensors. The software and hardware of the car's dashboard must process several input sensor data, such as the current speed at which the car is moving. All three sensors presented in the use case are independent of each other and non-redundant. But the unit they measure is of different safety-related criticality and therefore they have different ASILs. A single dashboard application software unit processes the input data from these three sensors. The problem is that according to ISO 26262 the entire software development of the dashboard must be carried out according to the highest ASIL of all input sensors. In this case, ASIL decomposition is effective and can lead to a significant reduction in time and cost due to the lowering of the ASIL of the implemented elements. In this use case, we first describe the ASIL tailoring using the SysML and the *ASIL SysML Profile* to ensure that all modeling elements are assigned the correct ASIL. Afterward, the decomposition of the dashboard application software is performed and described step by step using the same technologies.

The first action is the item definition at the block level to define a *safetyBlock*. This *safetyBlock* represents the system element for the car's dashboard and defines the boundary to other blocks. The *safetyBlock* named *Dashboard (Sensor Part)* is composed of property elements defined during the second action, namely the item definition at the property level. The dashboard system is composed of three hardware elements and one software element. More precisely, the *hardwareSafetyProperty: RPM Sensor*, *hardwareSafetyProperty: Speed Sensor*, *hardwareSafeytyProperty: Temperature Sensor*, the *softwareSafetyProperty: Dashboard Software*, and their corresponding ports for the data flow. The *Internal Block Diagram* illustrated in Figure 4.12 describes the *safetyBlock: Dashboard* in orange and its hardware & software safety properties and their ports in blue color. The safety requirements represented in green are defined at a later stage.

After the item definition at the block and property level has been carried out, the Hazard Analysis and Risk Assessment can be performed to define corresponding safety goals for each safety property. The safety goals and safety requirements are modeled in the *Requirement Diagram* illustrated in Figure 4.13. A *safetyGoal* element is defined for and assigned to each (dependency: *assignedTo*) of the three sensors and to the dashboard

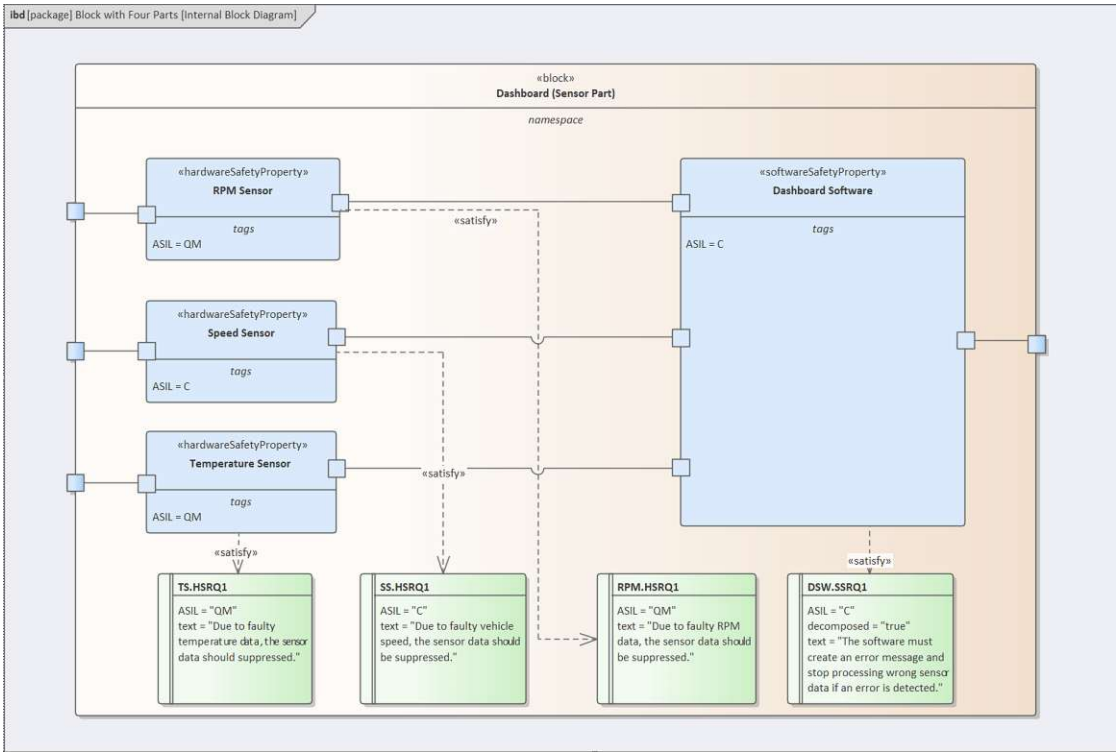

Figure 4.12: *Internal Block Diagram* before decomposition.

software. The *safetyGoal* attributes *con* (controllability), *exp* (probability of exposure) and *sev* (severity) are determined by a systematic evaluation during the Hazard Analysis and Risk Assessment. Afterward, the *ASIL* is determined for each safety goal according to these three attributes.[22] The three *hardwareSafetyProperty* elements and the *softwareSafetyProperty* element now each address a separately assigned *safetyGoal*.

The next step is to create the hierarchy and structure of safety requirements. For each *safetyGoal* element a *safetyRequirement* and afterward a *functionalSafetyRequirement* are derived. The safety requirement hierarchy is traced by the dependency *deriveReqt*. Subsequently, the *softwareSafetyRequirement: DSW.SSRQ1* is derived from its corresponding *functionalSafetyRequirement: DSW.FSRQ1* to specify ASIL-dependent requirements for software development. The *softwareSafetyRequirement: DSW.SSRQ1* is satisfied by the *softwareSafetyProperty: Dashboard Software*. The derivation of *hardwareSafetyRequirement* elements is carried out in the same way. For instance, the *hardwareSafetyRequirement: RPM.HSRQ1* is derived from its corresponding *functionalSafetyRequirement: RPM.FSRQ1* to specify ASIL-dependent requirements for hardware development. Afterward, the *hardwareSafetyProperty: RPM Sensor* is linked with the dependency *satisfy* to the *hardwareSafetyRequirement: RPM.HSRQ1*. The same steps have also been per-

[22] The ASIL determination table is listed in [ISOg, p. 10].

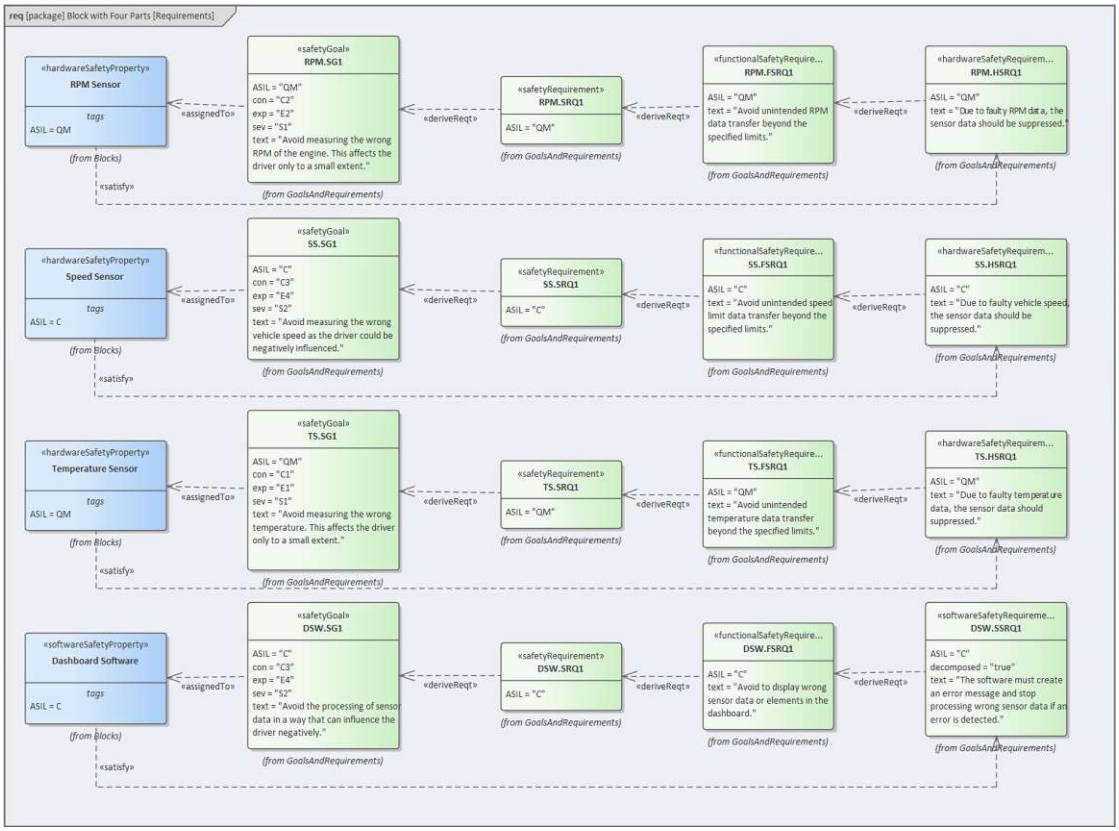

Figure 4.13: *Requirement Diagram* before decomposition.

formed for the two other sensors, more precisely for the *hardwareSafetyProperty: Speed Sensor* and the *hardwareSafetyProperty: Temperature Sensor*. Up to this point, each safety property has determined an ASIL, a linked safety goal, and a hardware-specific or software-specific safety requirement it satisfies. The specific safety requirements are represented in green in Figure 4.12.

The *Internal Block Diagram* and the *Requirement Diagram* are now fully modeled in compliance with ISO 26262. Unfortunately, the development of the dashboard software is facing a serious problem. According to ISO 26262, the entire dashboard's software development must be carried out according to the highest ASIL of all input sensors. In this case, the high ASIL is determined by the *Speed Sensor* with the ASIL C. The *RPM Sensor* and *Temperature Sensor* only have the ASIL QM. In order to save development costs and time, we lower the ASIL of the dashboard software through the technique of ASIL decomposition during the software design phase.

The ASIL decomposition is applied to the *softwareSafetyRequirement: DSW.SSRQ1* and its implementing *softwareSafetyProperty: Dashboard Software*. The performed decomposition is illustrated in Figure 4.14 and described subsequently.

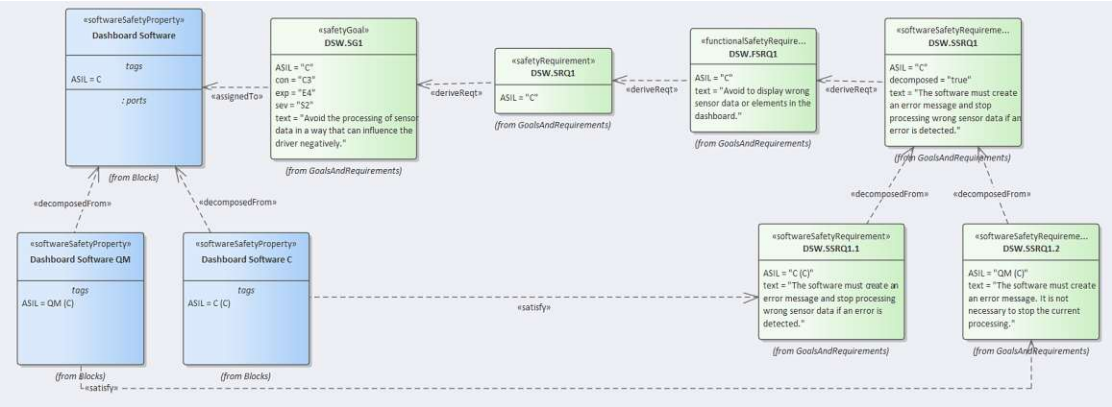

Figure 4.14: Relevant parts of the *Requirement Diagram* after decomposition.

The *softwareSafetyRequirement: DSW.SSRQ1* must be decomposed according to the ASIL decomposition schema stated in [ISOm, p. 8]. During ASIL decomposition two redundant safety requirements are created, more precisely the *softwareSafetyRequirement: DSW.SSRQ1.1* with the ASIL C (C) and the *softwareSafetyRequirement: DSW.SSRQ1.2* with the ASIL QM (C). The implementing software property element must also be decomposed to satisfy the newly created software safety requirements. The *softwareSafetyProperty: Dashboard Software* with the ASIL C is also decomposed into two redundant elements: the *softwareSafetyProperty: Dashboard Software QM* with the ASIL QM that processes the *RPM Sensor* and *Temperature Sensor* data as well as the *softwareSafetyProperty: Dashboard Software C* with the ASIL C that processes the *Speed Sensor* data. **This software level ASIL decomposition is acceptable only if the criteria of independence are verified by the analysis of dependent failures**[23]. In this use case, it is assumed that the criterion of independence is fulfilled.

In the *Internal Block Diagram* shown in Figure 4.15, the *softwareSafetyProperty: Dashboard Software* is now replaced by the decomposed *softwareSafetyProperty: Dashboard Software QM* and *softwareSafetyProperty: Dashboard Software C*. These two decomposed software property elements furthermore individually satisfy the *softwareSafetyRequirement: DSW.SSRQ1.1* and *softwareSafetyRequirement: DSW.SSRQ1.2* that are both decomposed from the *softwareSafetyRequirement: DSW.SSRQ1*. The part of the dashboard software processing the *RPM Sensor* and *Temperature Sensor* data can now be developed meeting only the requirements according to the ASIL QM. The other software

[23]The analysis of dependent failures is a complex task and not part of this work. It is described in [ISOm, p. 11-14]

part responsible for *Speed Sensor* data processing must be developed under much higher requirements according to the ASIL C. In summary, this can lead to significant time and cost savings during the software development.[24]

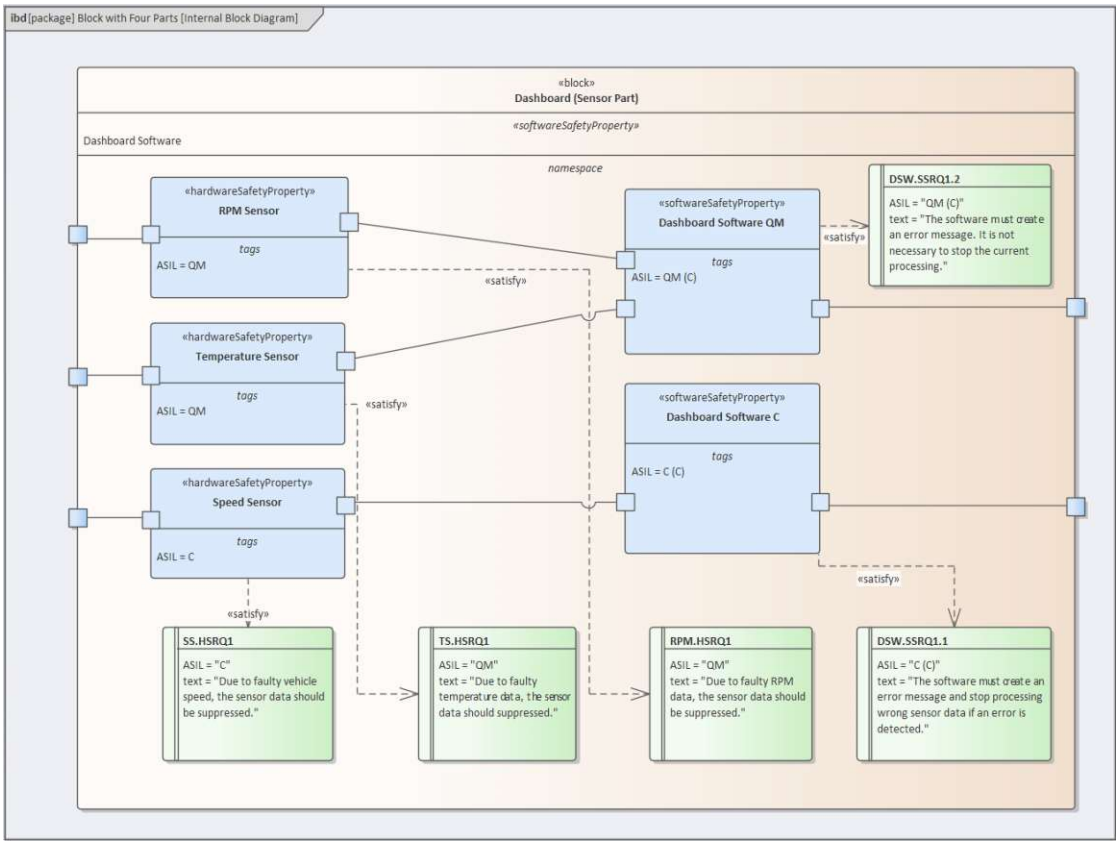

Figure 4.15: *Internal Block Diagram* after decomposition.

The presented use case has shown how the *ASIL SysML Profile* DSL can be used for model-based ASIL tailoring and decomposition with the partial example of the systems development of a car dashboard. All steps have been performed in Enterprise Architect Ultimate Version 15, SysML Version 1.5, and *ASIL SysML Profile* Version 1.0.

4.4 Evaluation

In the previous section 4.3, the *ASIL SysML Profile* DSL has been developed systematically. Furthermore, its utility has been shown by applying and modeling ASIL tailoring and decomposition using a real-life example of a SysML car dashboard systems devel-

[24] Note: The criteria of independence must always be verified again when further ASIL decomposition is performed!

opment. Apart from that, most importantly it is necessary to prove that the DSL is compliant with the relevant parts and clauses in the context of ISO 26262. According to Hevner et al. [HRM+04, p. 86], informed arguments can be used as a descriptive evaluation method to build a convincing argument for the created artifact's utility by using information from the knowledge base. For the creation process of the DSL we needed several different information sources and included them in the knowledge base (see Figure 4.6). However, for the descriptive evaluation performed in this section the focus is on the ISO 26262 series of standards for creating evaluation criteria. In the following subsection 4.4.1, we list the individual criteria for ASIL tailoring and decomposition specified in ISO 26262 and how they are fulfilled by the *ASIL SysML Profile* DSL. Each criterion and the associated evidence that this criterion has been met can be considered as an informed argument.

4.4.1 Evaluation Criteria

Criterion 1 - The requirements of the item shall be made available [ISOg, p. 4]: This criterion covers many different aspects such as performance or national legal requirements specified in the standard. However, we only consider the safety-related aspects, more precisely the safety-related requirements. These requirements can be made available by modeling *safetyRequirement*, *functionalSafetyRequirement*, *hardwareSafetyRequirement*, and *softwareSafetyRequirement* elements. Safety-related requirements can be classified after safety goals have been defined and their respective ASILs have been determined. This is ensured by deriving (dependency *deriveReqt*) the *safetyRequirement* from the *safetyGoal*. Furthermore, the correct derivations from *safetyRequirement* elements down to *softwareRequirement* and *hardwareSafetyRequirement* elements are ensured.

The constraints SRQ_2, SRQ_3, and SRQ_5 ensure that the modeled safety requirements are made available compliant with the ISO 26262 structure and dependencies of safety requirements.
The constraints SRQ_1, SRQ_11, SG_2, SG_3, SG_4, SG_5, SG_6, and SP_1 ensure the required ASIL consistency.

Criterion 2 - The elements of the item and its boundaries, its interfaces, and the assumptions concerning its interaction with other items and elements shall be defined [ISOg, p. 5]:
Safety-related elements can be modeled with the *safetyBlock* stereotype. The items of an element can be distinguished by functional-safety (*functionalSafetyProperty*), software-safety (*softwareSafetyProperty*), and hardware-safety (*hardwareSafetyProperty*) concerns. The safety properties and their safety block automatically define the border for safety concerns. SysML *ports* and *flow* properties define the interfaces and interaction between elements and blocks. SysML *ports* define the interface and are points at which external entities can connect to and interact with a block in different ways. The SysML *flow* properties define the interaction and specify the kinds of items that flow between blocks (e.g. data used for interaction).

The constraints SG_2, SP_1, SP_4 and SP_5 ensure consistent safety-related element and item definition. More precisely, the conditions cover the correct definition of the ASIL at all costs.

Criterion 3 - The Hazard Analysis and Risk Assessment shall be based on the item definition [ISOg, p. 6]:
Criterion 2 has already covered the definition of safety-related items. Thus, the Hazard Analysis and Risk Assessment can use the defined safety-related items as input. The Hazard Analysis and Risk Assessment is not explicitly modeled in our approach.[25] Safety goals are generated directly after the item definition and Hazard Analysis and Risk Assessment. The modeling of this process would be costly in terms of time and is furthermore not relevant for the ASIL tailoring and decomposition method constructed in this work. Thus, artifacts such as hazards and faults are not explicitly modeled.

Criterion 4 - The severity of potential harm shall be estimated based on a defined rationale for each hazardous event. The severity shall be assigned to one of the severity classes S0, S1, S2, or S3 [ISOg, p. 7]:
A risk assessment must be carried out for all hazardous events that are in the scope of ISO 26262. Hazardous events cannot be modeled with the developed *ASIL SysML Profile* DSL because this work does not focus on a model-based approach for the Hazard Analysis and Risk Assessment. A structured and model-based Hazard Analysis and Risk Assessment method for ISO 26262 has already been presented by Beckers et al. [BHFH13]. But since the severity classification is important for the ASIL determination of the corresponding safety goal, the attribute *severity* has been included in the stereotype *safetyGoal*. This improves the traceability of the ASILs' determination. The values for the attribute *severity* have been defined in an enumeration to avoid errors.

Criterion 5 - The probability of exposure of each operational situation shall be estimated based on a defined rationale for each hazardous event. The probability of exposure shall be assigned to one of the probability classes E0, E1, E2, E3, or E4 [ISOg, p. 8]:
The justification for meeting this criterion is the same as for criterion 4. The severity and exposure classes both only fulfill the purpose of ASIL determination in the presented model-based systems engineering process.

Criterion 6 - The controllability of each hazardous event by the driver or other persons involved in the operational situation shall be estimated based on a defined rationale for each hazardous event. The controllability shall be assigned to one of the controllability classes C0, C1, C2, or C3 [ISOg, p. 9]:
The justification for meeting this criterion is the same as for criterion 4. The severity

[25] A structured and model-based Hazard Analysis and Risk Assessment method for ISO 26262 is presented in [BHFH13].

and controllability classes both only fulfill the purpose of ASIL determination in the presented model-based systems engineering process.

Criterion 7 - A safety goal shall be determined for each hazardous event with an ASIL evaluated in the Hazard Analysis and Risk Assessment [ISOg, p. 10]:
A safety goal can be modeled with the modeling element *safetyGoal*. A *safetyGoal* is modeled and assigned to (dependency *assignedTo*) a *safetyProperty* or an element that specializes the stereotype *safetyProperty*. In the *ASIL SysML Profile* DSL, hazardous events are not explicitly modeled because the Hazard Analysis and Risk Assessment is not within the scope of this work. Thus, a *safetyGoal* element is not explicitly derived from a hazardous event. However, if this is necessary, the model-based Hazard Analysis and Risk Assessment method of Beckers et al. [BHFH13] can be applied between the item and safety goal definition introduced in this chapter.

The constraints SG_1, SG_2, and SG_3 ensure the ISO 26262 compliant ASIL definition of *safetyGoal* elements and the assignment of *safetyProperty* elements or its specializing stereotypes right after the *safetyGoal* definition.

Criterion 8 - An ASIL shall be determined for each hazardous event based on the classification of severity, probability of exposure, and controllability [ISOg, p. 9]:
The classification of severity, probability of exposure, and controllability is included in the stereotype *safetyGoal*. According to [ISOg], each hazardous event determines a *safetyGoal* element. But the method proposed in this work skips the Hazard Analysis and Risk Assessment. The modeling of the Hazard Analysis and Risk Assessment process is not required for the modeling of ASIL decomposition (see criterion 7) and can be added by including the approach of Beckers et al. [BHFH13].

The constraints SG_4, SG_5, and SG_6 ensure the classification of severity, probability of exposure, and controllability to determine the ASIL of a *safetyGoal*.

Criterion 9 - At least one functional safety requirement shall be derived from each safety goal [ISOg, p. 14]:
In the *ASIL SysML Profile* DSL, generic *safetyRequirement* elements are derived from *safetyGoal* elements in the first step. In the second step, the *functionalSafetyRequirement* elements are derived from the more generic *safetyRequirement* elements. This means that the proposed approach has one more step than necessary in the context of ISO 26262. This additional step helps to describe a requirement more generally in the first step before it is broken down at the functional, software or hardware level.

The constraint SRQ_2 ensures that a *safetyRequirement* must be derived from a *safetyGoal*. Furthermore, the constraint SRQ_3 ensures that a *functionalSafetyRequirement*

must be derived from a *safetyRequirement*.

Criterion 10 - The ASIL, as an attribute of the safety goal, is inherited by each subsequent safety requirement [ISOm, p. 4]:
The ASIL is first defined for *safetyGoal* elements. Afterward, the ASIL is inherited to all derived requirements at the functional, software and hardware level, more precisely to *safetyRequirement*, *functionalSafetyRequirement*, *softwareSafetyRequirement*, *hardwareSafetyRequirement* elements, and their corresponding subsequent derivations. This criterion does not cover any ASIL inheritance specifics when ASIL decomposition is applied. These specific inheritance characteristics are covered by criterion 20.

The ISO 26262 compliant ASIL inheritances across the top, functional, software and hardware level are ensured by the constraint SRQ_11.

Criterion 11 - During requirement allocation, the ASIL shall be inherited from the associated safety goal [ISOg, p. 15]:
The safety requirements shall be allocated to the elements of the system. In this approach, the safety requirements are allocated to safety properties. Thus, the defined safety properties must have the same ASIL as the allocated safety requirements. Furthermore, the inheritance of ASILs from *safetyGoal* elements to *safetyRequirement* elements is covered in criterion 10. The requirement allocation can be carried out at the functional, software or hardware level. This means that *functionalSafetyRequirements* can be allocated to *functionalSafetyProperties*, *softwareSafetyRequirements* to *softwareSafetyProperties*, or *hardwareSafetyRequirements* to *hardwareSafetyProperties*.

The ASIL inheritance during requirement allocation is ensured by the constraints SRQ_11, SRQ_12, SG_2, SP_4, SP_5, and SP_6.

Criterion 12 - The hardware safety requirements shall be allocated to the hardware elements. As a result, each hardware element shall be developed in compliance with the highest ASIL of any of the requirements allocated to it [ISOi, p. 6]:
Hardware elements are *hardwareSafetyProperty* elements in the *ASIL SysML Profile* DSL. A *hardwareSafetyProperty* element satisfies (dependency *satisfy*) one or more *hardwareSafetyRequirement* elements.

The constraint SRQ_12 ensures that the hardware element is developed with (at least) the same ASIL as the highest ASIL of any hardware requirements allocated.

Criterion 13 - If a hardware element is made of sub-elements that have an ASIL lower than the ASIL of the element or no ASIL assigned, then each of these shall be treated in accordance with the highest ASIL [ISOi, p. 10]:
All modeled *safetyProperty*, *functionalSafetyProperty*, *hardwareSafetyProperty*, and *soft-*

wareSafetyProperty elements can be nested. Thus, the modeling of hardware sub-elements as nested *hardwareSafetyProperty* elements is possible.

The constraints SRQ_12 and SP_7 ensure the correct ASIL assignment of sub-element modeling at the hardware level.

Criterion 14 - The software safety requirements are derived directly from the technical safety requirements allocated [ISOj, p. 8]**. A hardware safety requirements specification for the hardware elements of the item shall be derived from the technical safety requirements allocated to the hardware** [ISOi, p. 6]:
Technical safety requirements are not explicitly modeled in the *ASIL SysML Profile* DSL. The ASIL tailoring and decomposition does not differ on the technical level from the functional, software and hardware level. Thus, *abstractTechnicalSafetyRequirement* elements are not modeled and a *softwareSafetyRequirement* or *hardwareSafetyRequirement* is directly derived from a *functionalSafetyRequirement*. This simplifies the modeling without affecting the ASIL tailoring and decomposition principles. In summary, this criterion is intentionally not met.

The constraints SRQ_3, SRQ_4, and SRQ_5 ensure the correct derivation of safety requirements across all levels according to the decision that the technical level is not explicitly modeled.

Criterion 15 - If the embedded software has to implement software components of different ASILs or safety-related and non-safety-related software components, then all of the embedded software shall be treated in accordance with the highest ASIL [ISOj, p. 14]:
This criterion for software sub-components of software systems is very similar to criterion 13 for hardware sub-elements. The created *ASIL SysML Profile* DSL can model nested safety properties of all types. Thus, the modeling of software sub-components as nested *softwareSafetyProperty* elements is possible.

The constraints SRQ_12 and SP_7 ensure the correct ASIL assignment during software sub-component modeling.

Criterion 16 - ASIL decomposition shall be performed by considering each initial safety requirement individually [ISOm, p. 5]:
The developed DSL is able to decompose requirements at the functional, hardware and software level. At each level, each safety requirement is modeled separately. Thus, ASIL decomposition can be performed for each initial safety requirement individually if the proposed model-based ASIL decomposition rules are followed. Unfortunately, it is not possible to check whether several safety requirements have been modeled together using a constraint.

Criterion 17: The initial safety requirement shall be decomposed to redundant safety requirements that shall be implemented by sufficiently independent elements [ISOm, p. 5]:
A *functionalSafetyRequirement*, a *hardwareSafetyRequirement*, and a *softwareSafetyRequirement* can be decomposed by creating two (or more) redundant safety requirements from the same stereotype. Furthermore, the attribute *decomposed* of the parent safety requirement must be assigned the value "true". The decomposed safety requirements are linked with the parent element with the dependency *decomposedFrom*. The decomposition of the implementing *safetyProperty* elements and their specializing stereotypes that contribute to satisfying the decomposed safety requirements is carried out in the same way. Thus, this criterion is met except its necessary prove of sufficient independence. The modeling of the analysis of dependent failures to provide independence is not part of the *ASIL SysML Profile* DSL.[26]

The constraint SRQ_6 ensures that the dependency *decomposedFrom* is linked correctly. SRQ_9 ensures that the attribute *decomposed* is assigned the right value and SRQ_10 meets criterion 17 in detail. It ensures the number of redundant child elements and that all involved elements must have the same stereotype.

Criterion 18: Each decomposed safety requirement shall comply with the initial safety requirement by itself [ISOm, p. 5]:
If a safety requirement decomposition is modeled with the *ASIL SysML Profile* DSL, the initial safety requirement is retained regardless of the established level. During the proposed decomposition process two (or more) decomposed safety requirements are created and assigned to the initial safety requirement. Thus, the evaluation of the compliance of the decomposed requirements with the initial safety requirement is ensured.

There are no specific constraints defined for this criterion. Nevertheless, this criterion is supported by all constraints that ensure the ISO 26262 compliant ASIL decomposition requirements modeling, more precisely the constraints SRQ_6, SRQ_7, SRQ_8, SRQ_9, and SRQ_10.

Criterion 19: If ASIL decomposition of an initial safety requirement results in the allocation of decomposed requirements to the intended functionality and an associated safety mechanism, then a safety requirement shall be allocated to the intended functionality and implemented applying the corresponding decomposed ASIL [ISOm, p. 6]:
In the *ASIL SysML Profile* DSL, both the implementing item (*functionalSafetyProperty*, *softwareSafetyProperty*, *hardwareSafetyRequirement*) and the assigned safety requirement (*functionalSafetyRequirement*, *softwareSafetyRequirement*, *hardwareSafetyRequirement*) element are decomposed into redundant sub-elements. The initial elements are always

[26] More information about the analysis of dependent failures can be found in [ISOm, p. 11-14].

retained. Furthermore, the decomposed safety requirement and its decomposed ASIL can be directly assigned to the intended functionality of the decomposed implementing item. Thus, this criterion is met because the decomposed sub-elements created during the ASIL decomposition can be implemented applying the corresponding decomposed ASIL.

The constraints SRQ_9, SRQ_10, and SP_8 ensure that the involved elements are correctly decomposed into child elements and correctly allocated (dependency *satisfy*).

Criterion 20: When applying ASIL decomposition to a safety requirement, a decomposition schema must be chosen in accordance with the ASIL before decomposition. The ASIL decomposition schemas are shown in Figure 4.16 [ISOm, p. 6]:
The decomposition schema specifies the decomposition of safety requirements with ASIL A to ASIL D. The application of the decomposition schema can lead to advantageous time and cost savings. During the proposed decomposition process two (or more) decomposed safety requirements are created and assigned (dependency *derivedFrom*) to the initial safety requirement. The initial safety requirement is retained regardless of its established level, and its ASIL keeps the initially assigned value. The decomposed safety requirements have an ASIL with parentheses according to the defined enumeration named *ASIL*. The process of keeping the initial safety requirement and creating new decomposed safety requirements with ASILs in parentheses meets this criterion in combination with the associated constraint SRQ_8.

The constraint SRQ_8 ensures that the ASIL decomposition fulfills the ASIL decomposition schema requirements as illustrated in Figure 4.16 and stated in [ISOm, p. 6f].

Criterion 21: When applying ASIL decomposition to a safety requirement, an ASIL decomposition may be applied more than once [ISOm, p. 6]:
ASIL decomposition can be applied more than once when modeling with the *ASIL SysML Profile* DSL. The safety requirement (*functionalSafetyRequirement*, *softwareSafetyRequirement*, *hardwareSafetyRequirement*) can be decomposed any number of times. It is just necessary to assign the value true to the attribute *decomposed* for every decomposed requirement. Furthermore, decomposed requirements are assigned to their parent level by the dependency *decomposedFrom*. The same applies to safety property (*functionalSafetyProperty*, *softwareSafetyProperty*, *hardwareSafetyProperty*) element decomposition. If ASIL decomposition is applied more than once, the requirement allocation step is skipped for these safety properties and safety requirement elements at intermediate level(s). This means that no *satisfy* dependency is assigned within intermediate levels.

Criterion 22: When applying ASIL decomposition to a safety requirement, each decomposed ASIL shall be marked by putting the ASIL of the safety goal in parentheses [ISOm, p. 6]:
The enumeration *ASIL* contains among others the standard ASILs *A*, *B*, *C*, *D*, and *QM*.

To meet this criterion, the enumeration has been extended by all necessary combinations to mark the ASIL of the corresponding safety goal in parentheses. The attribute *ASIL* is included in all safety-related stereotypes in the *ASIL SysML Profile* DSL. But only safety-related modeling elements that can be decomposed[27] must have the ASILs for decomposition marked by putting the ASIL of the assigned safety goal in parentheses.

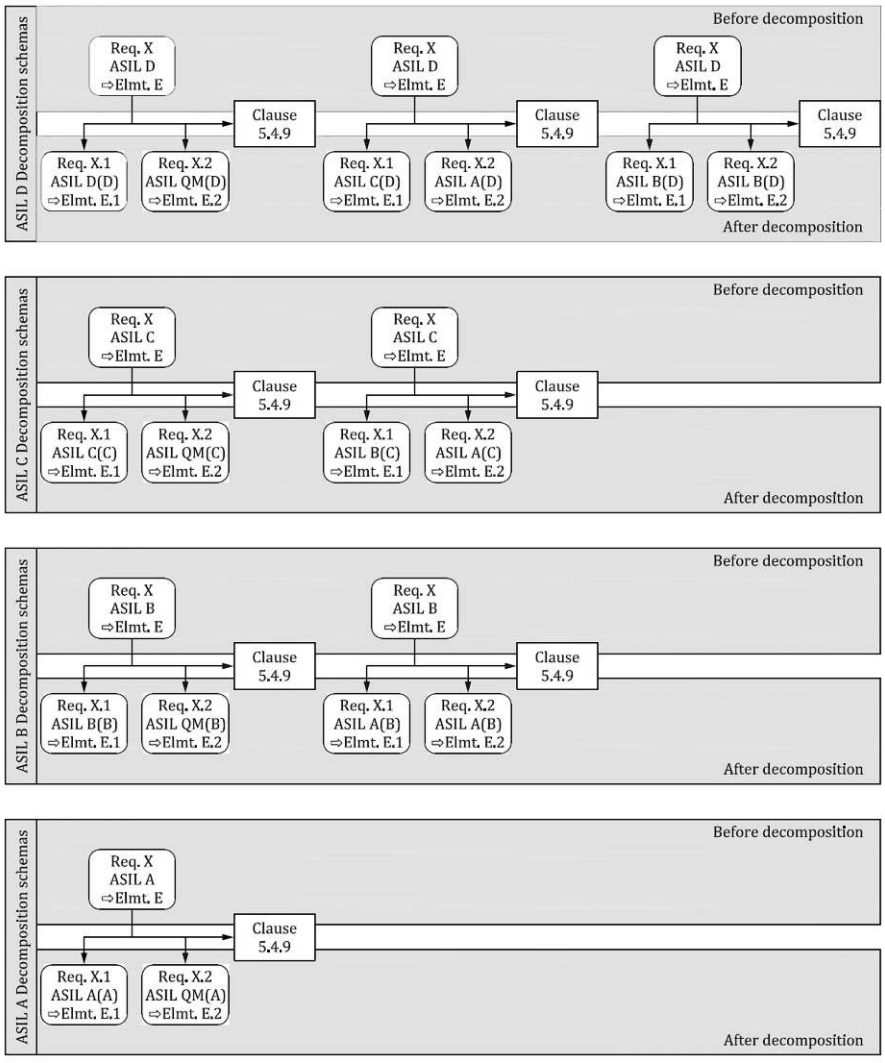

Figure 4.16: ASIL decomposition schema according to RQ 5.4.8 & RQ 5.4.9 (see table 4.1) [ISOm, p. 8].

For instance, the requirement X with an ASIL D is decomposed into two requirements

[27] Decomposable: *functionalSafetyProperty*, *hardwareSafetyProprety*, *softwareSafetyProperty*, *functionalSafetyRequirement*, *softwareSafetyRequirement*, *hardwareSafetyRequirement*.

Y and Z. According to the decomposition schema introduced in criterion 20, a possible decomposition is to assign the ASIL D (D) to the requirement Y and the ASIL QM (D) to the requirement Z. It is very important that decomposed safety requirements use the right type of ASIL. More precisely, the right choice between an ASIL with or without parentheses must be made.

The constraint SP_2 ensures that a decomposed element marks the attribute ASIL by putting the ASIL of the parent element (e.g. *safety goal*) in parentheses.

4.4.2 Evaluation Results

The descriptive evaluation carried out covered 22 informed arguments. All 22 criteria listed above are met and, thus, the created DSL, entitled *ASIL SysML Profile* can be used for ISO 26262 compliant modeling of ASIL tailoring and decomposition. The only exception is that a few criteria require the modeling of the Hazard Analysis and Risk Assessment. The solution in this work has been to include the required work products generated during the Hazard Analysis and Risk Assessment in other modeling elements. In addition, the created DSL could be combined with Beckers' [BHFH13] structured and model-based Hazard Analysis and Risk Assessment for ISO 26262 to create an even more sophisticated SysML extension. This has not been done for this work as their approach is not based on SysML and additional development effort would be necessary. Furthermore, the modeling of the Hazard Analysis and Risk Assessment does not influence the ASIL decomposition and thereby our research goals in any means.

It is important to note that only the performed final evaluation has been described here. The systematic development process of the DSL was characterized by a constant alternation between evaluation and development as it is recommended for IS research in [HRM+04].

4.5 Summary

The goal of this chapter has been to define an appropriate means to represent the ISO 26262 ASIL decomposition concept by extending the SysML. The development process has used the design-science framework from Hevner et al. [HRM+04] and has followed the guidelines for the systematic development of DSLs presented in [SZ09]. Furthermore, the MDG technology from [Spaa] has been used to extend the SysML with a SysML profile.

The theoretical approaches of ASIL tailoring and decomposition have been introduced at the beginning of this chapter. This was important because in order to find an appropriate means of applying ASIL decomposition with model-based engineering, we first had to facilitate the possibility for ASIL tailoring. ASIL tailoring is responsible for determining a classified ASIL for each safety-related element and must be completed before applying ASIL decomposition. The following systematic development of the DSL has been com-

prised of four steps. After these four steps have been carried out, the *ASIL SysML Profile* has been successfully created and applied in a use case that covers the development of an ISO 26262 compliant car dashboard. The use case has showed how useful the DSL is for applying ASIL decomposition in a model-based systems development process.

During the systematic development several iterations between development and evaluation phases have been carried out. A descriptive evaluation has been performed evaluating 22 different criteria. These criteria have been based on several ISO 26262 parts and have been analyzed to build convincing arguments to prove ISO 26262 compliance. At the end of the chapter the final evaluation has been conducted and described. All 22 criteria have been met by the *ASIL SysML Profile* and ensured by its OCL statements.

In summary, we have shown that the ASIL decomposition concept specified in part nine of ISO 26262 can be applied in model-based systems engineering. The created *ASIL SysML Profile* DSL can be used as a stand-alone SysML extension or in combination with other ISO 26262 model-based systems engineering approaches. For instance, this approach could be extended with the model-based Hazard Analysis and Risk Assessment method presented in [BHFH13]. This would only require the creation of a composite SysML profile.

CHAPTER 5

Conclusion

In this thesis, the paradigm shift from classical document-centric approaches to model-based engineering for systems development was discussed. This paradigm shift is currently in process and entails many advantages to manage the sophisticated requirements for safety standard compliance during systems development such as for self-driving cars. However, current solutions are not yet fully mature and many aspects still need to be researched. Two of these problems were addressed in this work. Firstly, that there is no structure and no big picture of the application of MBSE in the development of functional safety systems—a problem that was solved in this thesis by means of a Systematic Mapping Study (SMS) to structure this field of interest. Secondly, how MBSE can be applied to facilitate safety standard compliant systems engineering. This issue was solved by extending SysML to facilitate partly ISO 26262 compliant MBSE.

Systematic Mapping Study

In the first part of the thesis, an SMS was conducted to investigate model-based engineering for the development of security-certified safety-critical systems. A total number of 351 relevant publications from different scientific sources was selected for the systematic mapping in the context of IEC 61508 for E/E/PE systems and its industry- and application-specific variants, more precisely the specific variants ISO 26262 for road vehicles, EN 50128 for railway systems, and DO-178 for airborne systems. The contribution of the results of the SMS can be structured into three parts:

1. The first research question investigated which of the more than ten different industry- and application-specific variants of IEC 61508 are applied most frequently in the context of model-based engineering. The results showed that the classified publications most frequently addressed ISO 26262, followed by DO-178 and EN 50128. These three most often addressed safety standards as well as IEC 61508 were analyzed in more detail in the second and third research question.

2. The second research question focused on the reasons for applying model-based methods. The goal was to find out whether the safety standards IEC 61508, ISO 26262, DO-178, and EN 50128 apply model-based methods for the same or different reasons. The results showed that the reasons are manifold and slightly differ between these four standards. Nevertheless, in summary, the reasons *Safety Analysis/Certification*, *System Development*, and *Traceability* were the most commonly stated reasons for applying model-based methods.

3. The third research question followed the same principles as the second research question and was intended to provide an overview of which model-based methods are applied for which safety standards. The results revealed that the characteristics of the applied model-based methods are almost identical to the reasons for using model-based methods. There are some differences in the methods applied between the four investigated safety standards, but overall he methods applied and their popularity are rather similar. The most commonly applied model-based methods are *Model Transformations*, *Meta-Model Modeling*, and *Code Generation*.

In summary, the results of this SMS showed that model-based engineering mostly concerns safety assurance, regardless of whether it addresses IEC 61508 or one of its industry- or application-specific variants. Furthermore, the systematic mapping process led to the impression that most pursued research outcomes did not reach the degree of maturity necessary for application in practice. Challenges such as tool support and tool qualification are issues that are in need of further work. In the end, the question arises whether for model-based development to become established, more standardization and guidance would probably be necessary in the functional safety standards' specifications with regard to model-based engineering. For instance, in addition to guidelines, a uniform meta-model or UML/SysML profile ought to be specified in the context of ISO 26262. The ANSI/ISA-95 is an international standard that has already successfully realized this feature.[1]

SysML Extension

In the second part of the thesis, a DSL was systematically developed to define an appropriate means to represent the ASIL tailoring and decomposition concepts in the SysML. The created DSL, entitled *ASIL SysML Profile*, extends the SysML and is partly compliant with the ISO 26262 safety life cycle concerning ASIL tailoring and decomposition. The SysML is defined as a UML 2 profile and therefore it is possible to extend SysML elements with the help of the *UML Profiles* package—an extension possibility which we have used for this purpose. The technology used for implementation is the Model Driven Generation (MDG) by Sparx Systems[2].

[1] See [MH15] for more information.
[2] See [Spaa].

The systematic development comprised four steps. The first step was to apply a generalization to SysML elements and to define OCL constraints to specifically support the ASIL tailoring and decomposition concepts. In the second step the behavior was defined using a UML Activity Diagram, and in the third step the graphical concrete syntax was defined. In the last step the DSL was integrated with the Enterprise Architect platform and published as an MDG that can be imported by any user with domain knowledge. Afterward, a use case was presented to show a simplified example of how the application of the ASIL decomposition concept can be represented and supported by the *ASIL SysML Profile*.

The descriptive evaluation based on 22 criteria proved the DSL's compliance with the relevant parts and clauses of ISO 26262. The systematic development carried out several iterations between development and evaluation in order to achieve the claimed research goal. Nevertheless, the developed DSL can only be regarded as a prototype and leaves room for some further improvements. Firstly, to ensure comprehensive utility for the modeling of ASIL tailoring, the modeling of the Hazard Analysis and Risk Assessment must be provided. A possible solution would be to integrate the model-based Hazard Analysis and Risk Assessment method proposed by Beckers et al. [BHFH13]. Secondly, model validation using OCL constraints is the preferred solution to provide tool-independent customizations of a meta-model. However, not all tools support fully featured OCL validation, e.g. Enterprise Architect, or this validation is simply not time-efficient. Accordingly, the OCL constraints must be transferred to tool-specific model checking mechanisms. For example, it is necessary to create a compliant C# validation script for the Enterprise Architect build-in validation or consider external OCL validation solutions. Apart from that, there are further ideas for improvement such as a specific graphical appearance of safety-related modeling elements. For instance, safety-related modeling elements with an assigned ASIL should be highlighted with a particular color in relation to their level.

In summary, the second part of this thesis showed that MBSE can be used for systems development compliant with a specific functional safety standard. More precisely, the developed DSL can represent the ASIL decomposition concept as specified in part nine of the ISO 26262 series of standards[3]. The created *ASIL SysML Profile* can be used as a stand-alone SysML extension but, more importantly, also in combination with other sophisticated MBSE approaches such as [BHFH13, BCF+14, BCF+15].

[3] See [ISOm].

List of Figures

List of Tables

Bibliography

[ACM66] volume 9 of *Communications of the ACM*, New York, NY, USA, 1966. ACM.

[Adl14] Nico Adler. *Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262*. KIT Scientific Publishing, 2014.

[ADTK12] Morayo Adedjouma, Hubert Dubois, François Terrier, and Tarek Kitouni. An experiment on merging quality assessment in automotive domain. volume 290, pages 107–117, 05 2012.

[Afl17] L. Aflord. Do-331 model based development and verification supplement to do178c and do-278a. *APT Research, Inc*, 2017.

[AGB13] A. Alturki, Guy Gable, and W. Bandara. The design science research roadmap: In progress evaluation. *Proceedings - Pacific Asia Conference on Information Systems, PACIS 2013*, 01 2013.

[AGK+14] Neil Audsley, I. Gray, D. Kolovos, N. Matragkas, R. Paige, and Leandro Indrusiak. Automatic development of embedded systems using model driven engineering and compile-time virtualisation. 520:23–53, 01 2014.

[AGMG11] Nico Adler, Philipp Graf, and Klaus Müller-Glaser. Model-based consistency checks of electric and electronic architectures against requirements. volume 7167, pages 262–275, 10 2011.

[AKP99] William Aspray, Reinhard Keil, and David Parnas. History of software engineering. 01 1999.

[Alt12] Oliver Alt. *Modellbasierte Systementwicklung mit SysML*. Carl Hanser Verlag München, 2012.

[Alt14a] Oliver Alt. Survive ISO 26262 with model-based development! *embedded world conference*, 02 2014.

[Alt14b] Oliver Alt. Uml, sysml und autosar erfolgreich kombinieren und gemeinsam einsetzen. Automotive Software Kongress, 2014.

[AMYL15] Rouwaida Abdallah, Anas Motii, Nataliya Yakymets, and Agnes Lanusse. Using model driven engineering to support multi-paradigms security analysis. volume 580, pages 278–292, 02 2015.

[ARP] ARP4754A:2010 Guidelines for development of civil aircraft and systems.

[ASLN] Isaac Amundson, Lyle A. Shipton, Anshuo Liu, and Michael Nowak. *Toward Efficient Model-Based Development of Aerospace Applications*.

[AZ13] Albert Albers and Christian Zingel. Extending sysml for engineering designers by integration of the contact & channel. *Procedia Computer Science*, 16:353–362, 12 2013.

[Bah97] Nicholas Bahr. *System Safety Engineering And Risk Assessment*. Taylor & Francis, 1997.

[BB19] John Borky and Thomas Bradley. *Effective Model-Based Systems Engineering*. Springer, 2019.

[BC07] Kitchenham BA and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2, 01 2007.

[BCBK17] Souvik Barat, Tony Clark, Balbir Barn, and Vinay Kulkarni. A model-based approach to systematic review of research literature. In *Proceedings of the 10th Innovations in Software Engineering Conference*, ISEC '17, pages 15–25, New York, NY, USA, 2017. ACM.

[BCF+14] Kristian Beckers, Isabelle Côté, Thomas Frese, Denis Hatebur, and Maritta Heisel. Systematic derivation of functional safety requirements for automotive systems. In Andrea Bondavalli and Felicita Di Giandomenico, editors, *Computer Safety, Reliability, and Security*, pages 65–80, Cham, 2014. Springer International Publishing.

[BCF+15] Kristian Beckers, Isabelle Côté, Thomas Frese, Denis Hatebur, and Maritta Heisel. A structured validation and verification method for automotive systems considering the oem/supplier interface. volume 9337, 09 2015.

[BCF+16] Kristian Beckers, Isabelle Côté, Thomas Frese, Denis Hatebur, and Maritta Heisel. A structured and systematic model-based development method for automotive systems, considering the oem/supplier interface. *Reliability Engineering & System Safety*, 158, 09 2016.

[BCH13] Guillaume Barbier, Véronique Cucchi, and David R. C. Hill. Contribution of model-driven engineering to crop modeling. In Beniamino Murgante, Sanjay Misra, Maurizio Carlini, Carmelo M. Torre, Hong-Quang Nguyen, David Taniar, Bernady O. Apduhan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications – ICCSA 2013*, pages 253–263, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[BCK+16] Hans Blom, De-Jiu Chen, Henrik Kaijser, Henrik Lönn, Yiannis Papadopoulos, Mark-Oliver Reiser, Ramin Tavakoli Kolagari, and Sara Tucci-Piergiovanni. East-adl:: An architecture description language for automotive software-intensive systems in the light of recent use and research. *International Journal of System Dynamics Applications*, 5:1–20, 07 2016.

[BCW17] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool Publishers, 2nd edition, 2017.

[BDMM+14] Gregorio Barberio, Beniamino Di Martino, Nicola Mazzocca, Luigi Velardi, Aniello Amato, Renato Guglielmo, Ugo Gentile, Stefano Marrone, Roberto Nardone, Adriano Peron, and Valeria Vittorini. An interoperable testing environment for ertms/etcs control systems. 09 2014.

[Bei10] Michael Beine. A Model-Based Reference Workflow for the Development of Safety-Critical Software. In *ERTS2 2010, Embedded Real Time Software & Systems*, Toulouse, France, May 2010.

[BFK10] Marc Born, John Favaro, and Olaf Kath. Application of iso dis 26262 in practice. In *Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness & Safety*, CARS '10, pages 3–6, New York, NY, USA, 2010. ACM.

[BFP15] Stephan Baumgart, Joakim Fröberg, and Sasikumar Punnekkat. Enhancing model-based engineering of product lines by adding functional safety. 09 2015.

[BH] James Bruck and Kenn Hussey. Customizing uml: Which technique is right for you? `https://www.eclipse.org/modeling/mdt/uml2/docs/articles/Customizing_UML2_Which_Technique_is_Right_For_You/article.html`. Accessed: 2020-03-27.

[BHFH13] Kristian Beckers, Maritta Heisel, Thomas Frese, and Denis Hatebur. A structured and model-based hazard analysis and risk assessment method for automotive systems. pages 238–247, 11 2013.

[BHG+12] Ottmar Bender, Martin Hiller, Maurice Girod, Carsten Strobel, Martin Waßmuth, and Laurent Dieudonné. *Application and Evaluation in the Avionics Domain*, pages 177–196. 10 2012.

[Bis16] Desmond Bisandu. Design science research methodology in computer science and information systems. *International Journal of Information Technology*, 11 2016.

[BJ18] Mouza Blooshi and Shafer Jafer. Review of formal agile methods as cost-effective airworthiness certification processes. *Journal of Aerospace Information Systems*, 15:1–14, 07 2018.

[BMR+15] Valentina Bonfiglio, Leonardo Montecchi, Francesco Rossi, Paolo Lollini, András Pataricza, and Andrea Bondavalli. Executable models to support automated software fmea. volume 2015, pages 189–196, 01 2015.

[BRB17] Harald Bucher, Clemens Reichmann, and Jürgen Becker. An integrated approach enabling cross-domain simulation of model-based e/e-architectures. 2017.

[BS12] Fabien Belmonte and Elie Soubiran. A model based approach for safety analysis. pages 50–63, 09 2012.

[BSK10] C. Buckl, D. Sojer, and A. Knoll. Ftos: Model-driven development of fault-tolerant automation systems. In *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pages 1–8, Sep. 2010.

[CAGUM18] Julieth Patricia Castellanos Ardila, Barbara Gallina, and Faiz Ul Muram. Transforming spem 2.0-compatible process models into models checkable for compliance. In Ioannis Stamelos, Rory V. O'Connor, Terry Rout, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 233–247, Cham, 2018. Springer International Publishing.

[Cam82] M. Campbell-Kelly. The development of computer programming in britain (1945 to 1955). *Annals of the History of Computing*, 4(2):121–139, April 1982.

[CDDS10] Barbara Czerny, Joseph D'Ambrosio, Rami Debouk, and Kelly Stashko. Iso 26262 functional safety draft international standard for road vehicles: Background, status and overview. In *ISSC*, Minneapolis, Minnesota, USA, 2010.

[CJL+11] De-Jiu Chen, Rolf Johansson, Henrik Lönn, Hans Blom, Martin Walker, Yiannis Papadopoulos, Sandra Torchiaro, Fulvio Tagliabo, and Anders Sandberg. Integrated safety and architecture modeling for automotive embedded systems*. *e & i Elektrotechnik und Informationstechnik*, 128:196–202, 2011.

[Cla09] J. O. Clark. System of systems engineering and family of systems engineering from a standards, v-model, and dual-v model perspective. In *2009 3rd Annual IEEE Systems Conference*, pages 381–387, March 2009.

[CM16] Eric Carroll and Robert Joseph Malins. Systematic literature review: How is model-based systems engineering justified?. 2016.

[Con09] Mirko Conrad. Testing-based translation validation of generated code in the context of iec 61508. *Form. Methods Syst. Des.*, 35(3):389–401, December 2009.

[Con12] Mirko Conrad. Verification and Validation According to ISO 26262: A Workflow to Facilitate the Development of High-Integrity Software. In *Embedded Real Time Software and Systems (ERTS2012)*, Toulouse, France, February 2012.

[DDAA15] Darina Dicheva, Christo Dichev, Gennady Agre, and Galia Angelova. Gamification in education: A systematic mapping study. *Educational Technology & Society*, 18:75–88, 07 2015.

[DGWC14] Jim Davies, Jeremy Gibbons, James Welch, and Edward Crichton. Model-driven engineering of information systems: 10 years and 1000 versions. *Science of Computer Programming*, 89:88–104, 09 2014.

[DHS15] Dirk Dürholz, Steffen Herrmann, and Ralf Stärk. *Safety Essentials: ISO 26262 At a Glance*. Kugler Maag, Kornwestheim, Germany, 2015.

[dlVMAG17] Jose Luis de la Vara, Beatriz Marín, Clara Ayora, and Giovanni Giachetti. An experimental evaluation of the understanding of safety compliance needs with models. In Heinrich C. Mayr, Giancarlo Guizzardi, Hui Ma, and Oscar Pastor, editors, *Conceptual Modeling*, pages 239–247, Cham, 2017. Springer International Publishing.

[dlVPW13] Jose Luis de la Vara and Rajwinder Kaur Panesar-Walawege. Safetymet: A metamodel for safety standards. In Ana Moreira, Bernhard Schätz, Jeff Gray, Antonio Vallecillo, and Peter Clarke, editors, *Model-Driven Engineering Languages and Systems*, pages 69–86, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[DO-a] DO-178B:1992 Software Considerations in Airborne Systems and Equipment Certification.

[DO-b] DO-178C:2012 Software Considerations in Airborne Systems and Equipment Certification.

[DO-c] DO-331:2011 Model-Based Development and Verification Supplement to DO-178C and DO-278A.

[DO-d] DO-332:2011 Object-Oriented Technology and Related Techniques Supplement to DO-178C and DO-278A.

[DO-e] DO-333:2011 Model-Based Development and Verification Supplement to DO-178C and DO-278A.

[Dor16] Dov Dori. *Model-Based Systems Engineering with OPM and SysML*. Springer, 2016.

[Dou06] Schmidt Douglas. Model-Driven Engineering. *IEEE Computer)*, 39(2):25–31, 2006.

[DS10] Pierre David and M. Shawky. Supporting iso 26262 with sysml, benefits and limits. *Proceedings of ESREL 2010*, 09 2010.

[EAV] Ea build-in model validation. `https://www.sparxsystems.com/enterprise_architect_user_guide/15.1/model_domains/zf_diagram_model_validation.html`. Accessed: 01-28-2020.

[Els] Elsevier (publisher). `https://www.elsevier.com/`. Accessed: 12-12-2019.

[EN:] European committee for standardization official. `https://www.cen.eu/about/`. Accessed: 02-12-2019.

[EN5] Din en 50128; vde 0831-128:2012–03: Railway applications - communications, signalling and processing systems - software for railway control and protection systems.

[Eri05] Clifton Ericson. *Hazard analysis techniques for system safety*. John Wiley & Sons, Hoboken, New Jersey, USA, 2005.

[Erk15] Tom Erkkinen. Model-based design of complex embedded systems using industry standards. *Business Architecture Innovation Summit*, 2015.

[ESD13] Raymond Estrada, Gen Sasaki, and Eric Dillaber. Best practices for developing do-178 compliant software using model-based design. 08 2013.

[FAS+15] Peter Folkesson, Fatemeh Ayatolahi, Behrooz Sangchoolie, Jonny Vinter, Mafijul Islam, and Johan Karlsson. Back-to-back fault injection testing in model-based development. In Floor Koornneef and Coen van Gulijk, editors, *Computer Safety, Reliability, and Security*, pages 135–148, Cham, 2015. Springer International Publishing.

[FGH+18] T. Frese, N. Gerber, D. Hatebur, I. Côté, and M. Heisel. *Functional Safety Processes and Advanced Driver Assistance Systems: Evolution or Revolution?*, pages 199–216. Springer Fachmedien Wiesbaden, Wiesbaden, 2018.

[Foc16] Markus Fockel. Asil tailoring on functional safety requirements. volume 9923, pages 298–310, 09 2016.

[FV04] Lidia Fuentes and Antonio Vallecillo. An introduction to uml profiles. *UPGRADE, The European Journal for the Informatics Professional*, 5, 01 2004.

[Gal14] B. Gallina. A model-driven safety certification method for process compliance. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 204–209, Nov 2014.

[GC15] Jesús Padilla Gaeta and Krzysztof Czarnecki. Modeling aerospace systems product lines in sysml. In *Proceedings of the 19th International Conference on Software Product Line*, SPLC '15, page 293–302, New York, NY, USA, 2015. Association for Computing Machinery.

[GCSTT12] Francis Gacenga, Aileen Cater-Steel, Mark Toleman, and W-G Tan. A proposal and evaluation of a design method in design science research. *The Electronic Journal of Business Research Methods*, 10:89–100, 01 2012.

[GD15] Emanuel Grant and Tanaya Datta. Roadmap to a do-178c formal model-based software engineering methodology. *Lecture Notes in Engineering and Computer Science*, 1:460–465, 03 2015.

[GGME16] Barbara Gallina, Elena Gómez-Martínez, and Clara Benac Earle. Deriving safety case fragments for assessing mbasafe's compliance with en 50128. In Paul M. Clarke, Rory V. O'Connor, Terry Rout, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 3–16, Cham, 2016. Springer International Publishing.

[GGvH+18] Sinem Getir, Lars Grunske, André van Hoorn, Timo Kehrer, Yannic Noller, and Matthias Tichy. Supporting semi-automatic co-evolution of architecture and fault tree models. *Journal of Systems and Software*, 142, 04 2018.

[GKR+17] Filippo Grazioli, Evgeny Kusmenko, Alexander Roth, Bernhard Rumpe, and Michael Wenckstern. Simulation framework for executing component and connector models of self-driving vehicles. 09 2017.

[GMRBE+18] Elena Gómez-Martínez, Ricardo J Rodríguez, Clara Benac-Earle, Leire Etxeberria, and Miren Illarramendi. A methodology for model-based verification of safety contracts and performance requirements. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 232(3):227–247, 2018.

[Gro17a] Object Management Group. Systems modeling language specification version 1.5, 2017.

[Gro17b] Object Management Group. Unified modeling language specification version 2.5.1, 2017.

[Gro19a] Object Management Group. Systems modeling language specification version 1.6, 2019.

[Gro19b] Object Management Group. Uml profile for modeling and analysis of real-time and embedded systems (marte), 2019.

[GVR02] R.L. Glass, I. Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Technology*, 44(8):491 – 506, 2002.

[Har15] Laura Hart. Introduction to model-based system engineering (mbse) and sysml. *Delaware Valley INCOSE Chapter Meeting*, 2015.

[HDHM06] Klaus Hörmann, Lars Dittmann, Bernd Hindel, and Markus Müller. *SPICE in der Praxis*. dpunkt.verlag, 2006.

[HGZG12] Brahim Hamid, Jacob Geisel, Adel Ziani, and David Gonzalez. Safety lifecycle development process modeling for embedded systems - example of railway domain. In Paris Avgeriou, editor, *Software Engineering for Resilient Systems*, pages 63–75, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[HH01] K. J. Hayhurst and C. M. Holloway. Challenges in software aspects of aerospace systems. In *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*, pages 7–13, Nov 2001.

[HIRK10] Ibrahim Habli, Ireri Ibarra, Roger Rivett, and Tim Kelly. Model-based assurance for justifying automotive functional safety. 2010.

[HJK+11] Andreas Holzer, Visar Januzaj, Stefan Kugele, Boris Langer, Christian Schallhart, Michael Tautschnig, and Helmut Veith. Seamless testing for models and code. In Dimitra Giannakopoulou and Fernando Orejas, editors, *Fundamental Approaches to Software Engineering*, pages 278–293, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[HMZ+19] Fazilat Hojaji, Tanja Mayerhofer, Bahman Zamani, Abdelwahab Hamou-Lhadj, and Erwan Bousse. Model execution tracing: a systematic mapping study. *Software & Systems Modeling*, 02 2019.

[HRM+04] Alan Hevner, Alan R, Salvatore March, Salvatore T, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28:75–106, 03 2004.

[HSBS19] Philipp Heisig, Jan-Philipp Steghöfer, Christopher Brink, and Sabine Sachweh. A generic traceability metamodel for enabling unified end-to-end traceability in software product lines. pages 2344–2353, 04 2019.

[HSDZ+09] Holger Höhn, Bernhard Sechser, Klaudia Dussa-Zieger, Richard Messnarz, and Bernd Hindel. *Software Engineering nach Automotive SPICE.* dpunkt.verlag, 2009.

[IECa] IEC 61499 Function Blocks - Part 1: Architecture, Edition 2.0.

[IECb] IEC 61508-1:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 1: General requirements.

[IECc] IEC 61508-2:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 2: Requirements for electrical/-electronic/programmable electronic safety-related systems.

[IECd] IEC 61508-3:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 3: Software requirements.

[IECe] IEC 61508-4:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 4: Definitions and abbreviations.

[IECf] IEC 61508-5:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 5: Examples of methods for the determination of safety integrity levels.

[IECg] IEC 61508-6:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 6: Guidelines on the application of parts 2 and 3.

[IECh] IEC 61508-7:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - part 7: Overview of techniques and measures.

[ISK+19] M. Z. Iqbal, H. Sartaj, M. U. Khan, F. Ul Haq, and I. Qaisar. A model-based testing approach for cockpit display systems of avionics. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 67–77, Sep. 2019.

[ISOa] ISO 14971:2019 Medical devices — Application of risk management to medical devices.

[ISOb] ISO 26262-10:2018 Road vehicles — functional safety — part 10: Guidelines on iso 26262.

[ISOc] ISO 26262-11:2018 Road vehicles — functional safety — part 11: Guidelines on application of iso 26262 to semiconductors.

[ISOd] ISO 26262-1:2018 Road vehicles — functional safety — part 1: Vocabulary.

[ISOe] ISO 26262-12:2018 Road vehicles — functional safety — part 12: Adaptation of iso 26262 for motorcycles.

[ISOf] ISO 26262-2:2018 Road vehicles — functional safety — part 2: Management of functional safety.

[ISOg] ISO 26262-3:2018 Road vehicles — functional safety — part 3: Concept phase.

[ISOh] ISO 26262-4:2018 Road vehicles — functional safety — part 4: Product development at the system level.

[ISOi] ISO 26262-5:2018 Road vehicles — functional safety — part 5: Product development at the hardware level.

[ISOj] ISO 26262-6:2018 Road vehicles — functional safety — part 6: Product development at the software level.

[ISOk] ISO 26262-7:2018 Road vehicles — functional safety — part 7: Production, operation, service and decommissioning.

[ISOl] ISO 26262-8:2018 Road vehicles — functional safety — part 8: Supporting processes.

[ISOm] ISO 26262-9:2018 Road vehicles — functional safety — part 9: Automotive safety integrity level (asil)-oriented and safety-oriented analyses.

[ISOn] ISO 26262:2018 Annex b (informative).

[ISOo] ISO About us. `https://www.iso.org/about-us.html`. Accessed: 2019-12-01.

[ISOp] ISO/PAS 19450:2015 Automation systems and integration — Object-Process Methodology.

[Jac12] Stephen Jacklin. Certification of safety-critical software under do-178c and do-278a. Jun 2012.

[JBB09] Frédéric Jouault, Jean Bézivin, and Mikaël Barbero. Towards an advanced model-driven engineering toolbox. *ISSE*, 5:5–12, 03 2009.

[JDA+18] Shafagh Jafer, Umut Durak, Hakan Aydemir, Richard Ruff, and Thorsten Pawletta. *Advances in Software Engineering and Aeronautics*, pages 87–102. 05 2018.

[JHÖ11] Per Johannessen, Öjvind Halonen, and Ola Örsmark. Functional safety extensions to automotive spice according to iso 26262. In Rory V. O'Connor, Terry Rout, Fergal McCaffery, and Alec Dorling, editors, *Software Process Improvement and Capability Determination*, pages 52–63, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[Kel04] Tim Kelly. The goal structuring notation – a safety argument notation. 2004.

[Kha18] Maged Khalil. Pattern libraries guiding the model-based reuse of automotive solutions. In Ferhat Khendek and Reinhard Gotzhein, editors, *System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering*, pages 85–104, Cham, 2018. Springer International Publishing.

[Kir11] Sascha Kirstan. *Kosten und Nutzen modellbasierter Entwicklung eingebetteter Softwaresysteme im Automobil*. Dr. Hut Verlag, 2011.

[KWS+15] O. Kovalenko, M. Wimmer, M. Sabou, A. Lüder, F. J. Ekaputra, and S. Biffl. Modeling automationml: Semantic web technologies vs. model-driven engineering. In *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–4, 09 2015.

[LBE+13] Yaping Luo, M. Brand, Luc Engelen, John Favaro, Martijn Klabbers, and Giovanni Sartori. Extracting models from iso 26262 for reusable safety assurance. volume 7925, pages 192–207, 06 2013.

[LBEK15] Yaping Luo, M. Brand, Luc Engelen, and Martijn Klabbers. A modeling approach to support safety assurance in the automotive domain. *Advances in Intelligent Systems and Computing*, 1089:339–345, 01 2015.

[Lev95] Nancy G. Leveson. *Safeware: System Safety and Computers*. ACM, New York, NY, USA, 1995.

[LEvdB14] Yaping Luo, Luc Engelen, and Mark van den Brand. Metamodel comparison and model comparison for safety assurance. In Andrea Bondavalli, Andrea Ceccarelli, and Frank Ortmeier, editors, *Computer Safety, Reliability, and Security*, pages 419–430, Cham, 2014. Springer International Publishing.

[LKB19] Yaping Luo, Arash Khabbaz, and Mark Brand. *Safety-Driven Development and ISO 26262*, pages 225–254. 07 2019.

[LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1):134–152, Dec 1997.

[LvdBEK14] Yaping Luo, Mark van den Brand, Luc Engelen, and Martijn Klabbers. From conceptual models to safety assurance. In Eric Yu, Gillian Dobbie, Matthias Jarke, and Sandeep Purao, editors, *Conceptual Modeling*, pages 195–208, Cham, 2014. Springer International Publishing.

[LWWC12] Philip Langer, Konrad Wieland, Manuel Wimmer, and Jordi Cabot. Emf profiles: A lightweight extension approach for emf models. *Journal of Object Technology*, 11(1):1–29, April 2012.

[Mah90] Michael S. Mahoney. The roots of software engineering. volume 3 of *CWI Quarterly*, pages 325–334, Princeton, New Jersey, USA, 1990.

[MAL+11] Roland Mader, Eric Armengaud, Andrea Leitner, Christian Kreiner, Quentin Bourrouilh, Gerhard Griessnig, Christian Steger, and Reinhold Weiss. Computer-aided pha, fta and fmea for automotive embedded systems. pages 113–127, 01 2011.

[MAT+18] Peter Munk, Andreas Abele, Eike Thaden, Arne Nordmann, Rakshith Amarnath, Markus Schweizer, and Simon Burton. Invited: Semi-automatic safety analysis and optimization. pages 1–6, 06 2018.

[MB15] Asif Muhammad and Shahid Bhatti. Model driven architecture. 04 2015.

[MDA] Mda proposal by the object management group (omg). `https://www.omg.org/cgi-bin/doc?omg/2000-11-05`. Accessed: 28-02-2020.

[MH15] Alexandra Mazak and Christian Huemer. From business functions to control functions: Transforming rea to isa-95. pages 33–42, 07 2015.

[MLL+17] Saad Mubeen, Harold Lawson, John Lundback, Mattias Galnander, and Kurt-Lennart Lundback. Provisioning of predictable embedded software in the vehicle industry: The rubus approach. pages 3–9, 05 2017.

[MPS11] Alois Mayr, Reinhold Plösch, and Matthias Saft. Towards an operational safety standard for software: Modelling iec 61508 part 3. pages 97 – 104, 05 2011.

[MPS14] Alois Mayr, Reinhold Plösch, and Matthias Saft. Objective safety compliance checks for source code. *36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings*, 05 2014.

[MSAA13] Ayoub Khan Mohammad, Saeed Saqib, Darwish Ashraf, and Abraham Ajith. *Embedded and Real Time System Development: A Software Engineering Perspective*. Springer, 2013.

[MSS18] Salome Maro, Jan-Philipp Steghöfer, and Miroslaw Staron. Software traceability in the automotive domain: Challenges and solutions. *Journal of Systems and Software*, 141, 03 2018.

[MVAVD18] Mehrdad Moradi, Bert Van Acker, Ken Vanherpen, and Joachim Denil. *Model-Implemented Hybrid Fault Injection for Simulink (Tool demonstrations)*, pages 71–90. 12 2018.

[MZHJ19] Hana Mkaouar, Bechir Zalila, Jérôme Hugues, and Mohamed Jmaiel. A formal approach to aadl model-based software engineering. *International Journal on Software Tools for Technology Transfer*, 03 2019.

[NBR+19] Harald Naunheimer, Bernd Bertsche, Joachim Ryborz, Wolfgang Novak, and Peter Fietkau. *Getriebesteuerung – Elektrik, Elektronik, Aktuatorik und Sensorik*, pages 663–697. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.

[OFD+17] Pedro Oliveira, André L. Ferreira, Daniel Dias, Tiago Pereira, Paula Monteiro, and Ricardo J. Machado. An analysis of the commonality and differences between aspice and iso26262 in the context of software development. In Jakub Stolfa, Svatopluk Stolfa, Rory V. O'Connor, and Richard Messnarz, editors, *Systems, Software and Services Process Improvement*, pages 216–227, Cham, 2017. Springer International Publishing.

[OMGa] Object management group model driven architecture (mda) guide rev 2.0. `https://www.omg.org/cgi-bin/doc?ormsc/14-06-01`. Accessed: 2020-02-28.

[OMGb] OMG Standardized profiles. `https://www.omg.org/spec/#Profile`. Accessed: 2019-12-22.

[oSEI07] International Council on Systems Engineering (INCOSE). *SYSTEMS ENGINEERING VISION 2020*. 2007.

[Par99] European Parliament. Resolution on the report from the commission to the council and the european parliament. *Official Journal of the European Communities*, 1999.

[PDA17] G. Philip, M. Dsouza, and V. P. Abidha. Model based safety analysis: Automatic generation of safety validation test cases. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–10, Sep. 2017.

[PE19] A. Paz and G. El Boussaidi. A requirements modelling language to facilitate avionics software verification and certification. In *2019 IEEE/ACM 6th International Workshop on Requirements Engineering and Testing (RET)*, pages 1–8, May 2019.

[Pel18] J. Peleska. Model-based avionic systems testing for the airbus family. In *2018 IEEE 23rd European Test Symposium (ETS)*, pages 1–10, May 2018.

[PEML10] Jean-François Pétin, Dominique Evrot, Gérard Morel, and Pascal Lamy. Combining sysml and formal models for safety requirements verification. *22nd International Conference on Software & Sys tems Engineering and their Applications*, 12 2010.

[PF11] Andea Piovesan and John Favaro. Experience with iso 26262 and asil decomposition. Automotive SPIN, 2011.

[PFMM08] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 17, 06 2008.

[Pie19] Krzysztof Pietrusewicz. Metamodelling for design of mechatronic and cyber-physical systems. *Applied Sciences*, 9:376, 01 2019.

[PK15] D. A. Prosvirin and V. P. Kharchenko. Model-based solution and software engineering environment for uav critical onboard applications. In *2015 IEEE International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)*, pages 312–315, Oct 2015.

[Pot12] Bill Potter. Complying with do-178c and do-331 using model-based design. *Aerospace Electronics and Avionics Systems Conference*, 12, 2012.

[PVK15] Kai Petersen, Sairam Vakkalanka, and Ludwik Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1 – 18, 2015.

[PWSB11] Rajwinder Panesar-Walawege, Mehrdad Sabetzadeh, and Lionel Briand. A model-driven engineering approach to support the verification of compliance to safety standards. pages 30–39, 11 2011.

[RFB12] A. L. Ramos, J. V. Ferreira, and J. Barceló. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(1):101–111, 2012.

[RFT16] Klaus Reichl, Tomas Fischer, and Peter Tummeltshammer. Using formal methods for verification and validation in railway. volume 9762, pages 3–13, 07 2016.

[Rie13] Leanna Rierson. *Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance*. CRC Press, 2013.

[Ros14] Hans-Leo Ross. *Funktionale Sicherheit im Automobil: ISO 26262, Systemengineering auf Basis eines Sicherheitslebenszyklus und bewährten Managementsystemen*. Carl Hanser Verlag, 2014.

[SAE] Society of automotive engineers (sae) international. `https://www.sae.org/about/`. Accessed: 04-12-2019.

[SB18] D. Sales and L. Buss Becker. Systematic literature review of system engineering design methods. In *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 213–218, Nov 2018.

[Sch10] Alexander Schloske. Funktional sicherheit und deren umsetzung nach iec 61508 / iso dis 26262. In *Konferenzband zur 18. FED-Konferenz*, pages 213–222, Berlin, 2010. Fraunhofer.

[Sch19] Stefan Schulte. Research methods: Systematic literature reviews. University Lecture, 2019.

[SCS11] Mehrdad Saadatmand, Antonio Cicchetti, and Mikael Sjödin. On the need for extending marte with security concepts. 03 2011.

[SD17] Miroslaw Staron and Darko Durisic. *AUTOSAR Standard*, pages 81–116. Springer International Publishing, Cham, 2017.

[SDG15] Marc Sango, Laurence Duchien, and Christophe Gransart. Component-based modeling and observer-based verification for railway safety-critical applications. In Ivan Lanese and Eric Madelaine, editors, *Formal Aspects of Component Software*, pages 248–266, Cham, 2015. Springer International Publishing.

[SG19] J. Schumann and K. Goseva-Popstojanova. Verification and validation approaches for model-based software engineering. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 514–518, Sep. 2019.

[SKB11] D. Sojer, A. Knoll, and C. Buckl. Synthesis of diagnostic techniques based on an iec 61508-aware metamodel. In *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, pages 59–62, June 2011.

[SKDS+19] Rick Salay, Sahar Kokaly, Alessio Di Sandro, Nick L. S. Fung, and Marsha Chechik. Heterogeneous megamodel management using collection operators. *Software & Systems Modeling*, Jun 2019.

[SMHK15] Harald Sporer, Georg Macher, Andrea Höller, and Christian Kreiner. Bidirectional crosslinking of system and software modeling in the automotive domain. pages 99–113, 09 2015.

[SOE+08] Andreas Svendsen, Gøran K. Olsen, Jan Endresen, Thomas Moen, Erik Carlson, Kjell-Joar Alme, and Øystein Haugen. The future of train signaling. In Krzysztof Czarnecki, Ileana Ober, Jean-Michel Bruel, Axel Uhl, and Markus Völter, editors, *Model Driven Engineering Languages and Systems*, pages 128–142, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Spaa] Sparx Systems Model Driven Generatoin (MDG) Technologies. `https://sparxsystems.com/enterprise_architect_user_guide/15.1/modeling/mdg_technologies.html`. Accessed: 2020-02-13.

[Spab] Sparx Systems Part Definition. https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_domains/part.html. Accessed: 2020-03-10.

[Spi89] J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice-Hall, Inc., USA, 1989.

[SS16] David J Smith and Kenneth Simpson. *The Safety Critical Systems Handbook*. Elsevier, 4 edition, 2016.

[Sta17] Miroslaw Staron. *Software Architectures: Views and Documentation*, pages 19–50. Springer International Publishing, Cham, 2017.

[SUN15] Pengfei SUN. *Model based system engineering for safety of railway critical systems*. Theses, Ecole Centrale de Lille, July 2015.

[Sys] Sysml forum. https://sysmlforum.com/sysml-faq/what-is-relation-between-sysml-and-uml.html. Accessed: 2020-02-27.

[Sys17] Sparx Systems. *MDG Technologies User Guide*. Enterprise Architect, 2017.

[SZ09] Mark Strembeck and Uwe Zdun. An approach for the systematic development of domain-specific languages. *Softw. Pract. Exper.*, 39(15):1253–1292, October 2009.

[TLDP15] Davide Taibi, Valentina Lenarduzzi, Laurent Dieudonné, and Christiane Plociennik. Towards a classification schema for development technologies: an empirical study in the avionic domain. *International Journal On Advances in Software*, 8, 01 2015.

[Uni98] European Union. Sixteenth annual report on monitoring the application of community law. *Official Journal of the European Communities*, pages 1–192, 1998.

[VK13] Timo Vepsäläinen and Seppo Kuikka. *Simulation-Based Development of Safety Related Interlocks*, pages 165–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[VRE18] Jose Vara, Alejandra Ruiz, and Huáscar Espinoza. Recent advances towards the industrial application of model-driven engineering for assurance of safety-critical systems. pages 632–641, 01 2018.

[WBCW19] Andreas Wortmann, Olivier Barais, Benoit Combemale, and Manuel Wimmer. Modeling languages in industry 4.0: An extended systematic mapping study. *Springer Verlag*, pages 1–28, 2019.

[WCEK17] Laurent Wouters, Stephen Creff, Emma Effa, and Ali Koudri. Collaborative systems engineering: Issues & challenges. pages 486–491, 04 2017.

[WLG09] Haifeng Wang, Shuo Liu, and Chunhai Gao. Study on model-based safety verification of automatic train protection system. *PACIIA 2009 - 2009 2nd Asia-Pacific Conference on Computational Intelligence and Industrial Applications*, 1, 11 2009.

[WMC+19] Sabine Wolny, Alexandra Mazak, Christine Carpella, Verena Geist, and Manuel Wimmer. Thirteen years of sysml: a systematic mapping study. *Software & Systems Modeling*, May 2019.

[WTI] Wissenschaftlich-technische informationen (wti) frankfurt. `https://www.wti-frankfurt.de/en/`. Accessed: 12-12-2019.

[YLJC14] Zhuoqun Yang, Zhi Li, Zhi Jin, and Yunchuan Chen. A systematic literature review of requirements modeling and analysis for self-adaptive systems. In Camille Salinesi and Inge van de Weerd, editors, *Requirements Engineering: Foundation for Software Quality*, pages 55–71, Cham, 2014. Springer International Publishing.

[ZBL11] Gregory Zoughbi, Lionel Briand, and Yvan Labiche. Modeling safety and airworthiness (rtca do-178b) information: conceptual model and uml profile. *Software & Systems Modeling*, 10(3):337–367, Jul 2011.

[ZTK18] Markus Zoppelt and Ramin Tavakoli Kolagari. *SAM: A Security Abstraction Model for Automotive Software Systems*, pages 59–74. 09 2018.

Appendix A

IEC 61508 - SMS Publications

IEC 61508 - Springer

SP001 Abdallah R., Motii A., Yakymets N., Lanusse A. (2015) Using Model Driven Engineering to Support Multi-paradigms Security Analysis. In: Desfray P., Filipe J., Hammoudi S., Pires L. (eds) Model-Driven Engineering and Software Development. MODELSWARD 2015. Communications in Computer and Information Science, vol 580. Springer, Cham

SP002 Vepsäläinen T., Kuikka S. (2013) Simulation-Based Development of Safety Related Interlocks. In: Pina N., Kacprzyk J., Filipe J. (eds) Simulation and Modeling Methodologies, Technologies and Applications. Advances in Intelligent Systems and Computing, vol 197. Springer, Berlin, Heidelberg

SP004 Hamid B., Percebois C. (2014) Model-Based Specification and Validation of Security and Dependability Patterns. In: Danger J., Debbabi M., Marion JY., Garcia-Alfaro J., Zincir Heywood N. (eds) Foundations and Practice of Security. FPS 2013. Lecture Notes in Computer Science, vol 8352. Springer, Cham

SP005 Di Alesio, S. & Sen, S. Softw Syst Model (2018) 17: 479. https://doi.org/10.1007/s10270-017-0585-x

SP006 Pröll R., Rumpold A., Bauer B. (2018) Applying Integrated Domain-Specific Modeling for Multi-concerns Development of Complex Systems. In: Pires L., Hammoudi S., Selic B. (eds) Model-Driven Engineering and Software Development. MODELSWARD 2017. Communications in Computer and Information Science, vol 880. Springer, Cham

SP007 Macher G., Armengaud E., Kreiner C. (2015) Integration of Heterogeneous Tools to a Seamless Automotive Toolchain. In: O'Connor R., Umay Akkaya M., Kemaneci K., Yilmaz M., Poth A., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2015. Communications in Computer and Information Science, vol 543. Springer, Cham

SP008 Hamid, B., Gürgens, S. & Fuchs, A. Innovations Syst Softw Eng (2016) 12: 109. https://doi.org/10.1007/s11334-015-0259-1

SP009 Bertolino, A., Calabro', A., Di Giandomenico, F. et al. Software Qual J (2018) 26: 1223. https://doi.org/10.1007/s11219-017-9393-3

SP010 Zoughbi, G., Briand, L. & Labiche, Y. Softw Syst Model (2011) 10: 337. https://doi.org/10.1007/s10270-010-0164-x

SP011 Buckl C., Knoll A., Schieferdecker I., Zander J. (2010) 10 Model-Based Analysis and Development of Dependable Systems. In: Giese H., Karsai G., Lee E., Rumpe B., Schätz B. (eds) Model-Based Engineering of Embedded Real-Time Systems. MBEERTS 2007. Lecture Notes in Computer Science, vol 6100. Springer, Berlin, Heidelberg

SP012 de la Vara J.L., Panesar-Walawege R.K. (2013) SafetyMet: A Metamodel for Safety Standards. In: Moreira A., Schätz B., Gray J., Vallecillo A., Clarke P. (eds) Model-Driven Engineering Languages and Systems. MODELS 2013. Lecture Notes in Computer Science, vol 8107. Springer, Berlin, Heidelberg

SP013 Huhn M., Hungar H. (2010) 8 UML for Software Safety and Certification. In: Giese H., Karsai G., Lee E., Rumpe B., Schätz B. (eds) Model-Based Engineering of Embedded Real-Time Systems. MBEERTS 2007. Lecture Notes in Computer Science, vol 6100. Springer, Berlin, Heidelberg

SP014 de la Vara J.L. et al. (2012) Towards a Model-Based Evolutionary Chain of Evidence for Compliance with Safety Standards. In: Ortmeier F., Daniel P. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2012. Lecture Notes in Computer Science, vol 7613. Springer, Berlin, Heidelberg

SP015 Luo Y., van den Brand M., Engelen L., Klabbers M. (2014) From Conceptual Models to Safety Assurance. In: Yu E., Dobbie G., Jarke M., Purao S. (eds) Conceptual Modeling. ER 2014. Lecture Notes in Computer Science, vol 8824. Springer, Cham

SP016 Svendsen A. et al. (2008) The Future of Train Signaling. In: Czarnecki K., Ober I., Bruel JM., Uhl A., Völter M. (eds) Model Driven Engineering Languages and Systems. MODELS 2008. Lecture Notes in Computer Science, vol 5301. Springer, Berlin, Heidelberg

SP019 Sannier N., Baudry B. (2014) INCREMENT: A Mixed MDE-IR Approach for Regulatory Requirements Modeling and Analysis. In: Salinesi C., van de Weerd I. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2014. Lecture Notes in Computer Science, vol 8396. Springer, Cham

SP021 Panesar-Walawege R.K., Skyberg Knutsen T., Sabetzadeh M., Briand L. (2011) CRESCO: Construction of Evidence Repositories for Managing Standards Compliance. In: De Troyer O., Bauzer Medeiros C., Billen R., Hallot P., Simitsis A., Van Mingroot H. (eds) Advances in Conceptual Modeling. Recent Developments and New Directions. ER 2011. Lecture Notes in Computer Science, vol 6999. Springer, Berlin, Heidelberg

SP022 Conrad, M. Form Methods Syst Des (2009) 35: 389. https://doi.org/10.1007/s10703-009-0082-0

SP025 Hamid B., Geisel J., Ziani A., Gonzalez D. (2012) Safety Lifecycle Development Process Modeling for Embedded Systems - Example of Railway Domain. In: Avgeriou P. (eds) Software Engineering for Resilient Systems. SERENE 2012. Lecture Notes in Computer Science, vol 7527. Springer, Berlin, Heidelberg

SP026 Domis D., Trapp M. (2008) Integrating Safety Analyses and Component-Based Design. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

SP027 Adedjouma M., Dubois H., Terrier F., Kitouni T. (2012) Merging the Quality Assessment of Processes and Products in Automotive Domain. In: Dieste O., Jedlitschka A., Juristo N. (eds) Product-Focused Software Process Improvement. PROFES 2012. Lecture Notes in Computer Science, vol 7343. Springer, Berlin, Heidelberg

SP029 Becker U. (2009) Model-Based Development of Medical Devices. In: Buth B., Rabe G., Seyfarth T. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2009. Lecture Notes in Computer Science, vol 5775. Springer, Berlin, Heidelberg

SP030 Sojer, D., Buckl, C. & Knoll, A. Comput Sci Res Dev (2015) 30: 21. https://doi.org/10.1007/s00450-011-0203-z

SP031 von der Beeck M., Braun P., Rappl M., Schröder C. (2003) Automotive UML. In: Lavagno L., Martin G., Selic B. (eds) UML for Real. Springer, Boston, MA

SP033 Luo Y., Engelen L., van den Brand M. (2014) Metamodel Comparison and Model Comparison for Safety Assurance. In: Bondavalli A., Ceccarelli A., Ortmeier F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8696. Springer, Cham

SP034 Pedroza G. (2019) Towards Safety and Security Co-engineering. In: Hamid B., Gallina B., Shabtai A., Elovici Y., Garcia-Alfaro J. (eds) Security and Safety Interplay of Intelligent Software Systems. CSITS 2018, ISSA 2018. Lecture Notes in Computer Science, vol 11552. Springer, Cham

SP035 Martin H., Tschabuschnig K., Bridal O., Watzenig D. (2017) Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges?. In: Watzenig D., Horn M. (eds) Automated Driving. Springer, Cham

SP037 Luo Y., Saberi A.K., den Brand M.. (2019) Safety-Driven Development and ISO 26262. In: Dajsuren Y., van den Brand M. (eds) Automotive Systems and Software Engineering. Springer, Cham

SP038 Noyer, A., Iyenghar, P., Engelhardt, J. et al. Software Qual J (2017) 25: 671. https://doi.org/10.1007/s11219-016-9323-9

SP040 Lawford M., Maibaum T., Wassyng A. (2010) Certification of Software-Driven Medical Devices. In: Margaria T., Steffen B. (eds) Leveraging Applications of Formal Methods, Verification, and Validation. ISoLA 2010. Lecture Notes in Computer Science, vol 6416. Springer, Berlin, Heidelberg

SP041 Luo Y., van den Brand M., Engelen L., Favaro J., Klabbers M., Sartori G. (2013) Extracting Models from ISO 26262 for Reusable Safety Assurance. In: Favaro J., Morisio M. (eds) Safe and Secure Software Reuse. ICSR 2013. Lecture Notes in Computer Science, vol 7925. Springer, Berlin, Heidelberg

SP043 Chen D. et al. (2008) Modelling Support for Design of Safety-Critical Automotive Embedded Systems. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

SP045 Iosif-Lazăr A.F., Schaefer I., Wąsowski A. (2014) A Core Language for Separate Variability Modeling. In: Margaria T., Steffen B. (eds) Leveraging Applications of Formal Methods, Verification and Validation. Technologies for Mastering Change. ISoLA 2014. Lecture Notes in Computer Science, vol 8802. Springer, Berlin, Heidelberg

SP047 Zimmer B., Bürklen S., Knoop M., Höfflinger J., Trapp M. (2011) Vertical Safety Interfaces – Improving the Efficiency of Modular Certification. In: Flammini F., Bologna S., Vittorini V. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2011. Lecture Notes in Computer Science, vol 6894. Springer, Berlin, Heidelberg

SP052 Tekaya M., Bennani M.T., Alagui M.A., Ahmed S.B. (2015) Aspect-Oriented Test Case Generation from Matlab/Simulink Models. In: Zamojski W., Mazurkiewicz J., Sugier J., Walkowiak T., Kacprzyk J. (eds) Theory and Engineering of Complex Systems and Dependability. DepCoS-RELCOMEX 2015. Advances in Intelligent Systems and Computing, vol 365. Springer, Cham

SP054 Trei M., Maro S., Steghöfer JP., Peikenkamp T. (2016) An ISO 26262 Compliant Design Flow and Tool for Automotive Multicore Systems. In: Abrahamsson P., Jedlitschka A., Nguyen Duc A., Felderer M., Amasaki S., Mikkonen T. (eds)

Product-Focused Software Process Improvement. PROFES 2016. Lecture Notes in Computer Science, vol 10027. Springer, Cham

SP056 Kornecki, A. & Zalewski, J. Innovations Syst Softw Eng (2009) 5: 149. https://doi.org/10.1007/s11334-009-0088-1

SP059 Fornari X. (2019) Battery Management System: From Safe Architecture Definition to System Simulation with Embedded Software. In: Langheim J. (eds) Electronic Components and Systems for Automotive Applications. Lecture Notes in Mobility. Springer, Cham

SP060 Beckers K., Côté I., Frese T., Hatebur D., Heisel M. (2015) A Structured Validation and Verification Method for Automotive Systems Considering the OEM/Supplier Interface. In: Koornneef F., van Gulijk C. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 9337. Springer, Cham

SP062 Gonschorek T., Zeller M., Höfig K., Ortmeier F. (2018) Fault Trees vs. Component Fault Trees: An Empirical Study. In: Gallina B., Skavhaug A., Schoitsch E., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2018. Lecture Notes in Computer Science, vol 11094. Springer, Cham

SP065 Beckers K., Côté I., Frese T., Hatebur D., Heisel M. (2014) Systematic Derivation of Functional Safety Requirements for Automotive Systems. In: Bondavalli A., Di Giandomenico F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8666. Springer, Cham

SP066 Schoitsch, E., Althammer, E., Lamedschwandner, K. et al. Elektrotech. Inftech. (2006) 123: 369. https://doi.org/10.1007/s00502-006-0368-5

SP067 Liggesmeyer P., Trapp M. (2015) Safety in der Industrie 4.0. In: Vogel-Heuser B., Bauernhansl T., ten Hompel M. (eds) Handbuch Industrie 4.0. Springer NachschlageWissen. Springer Vieweg, Berlin, Heidelberg

SP074 Reif K. (2011) Software. In: Reif K. (eds) Bosch Autoelektrik und Autoelektronik. Vieweg+Teubner

SP075 Schnieder E., Schnieder L. (2013) Technische Entwicklung. In: Verkehrssicherheit. VDI-Buch. Springer Vieweg, Berlin, Heidelberg

SP076 Schäuffele J., Zurawka T. (2016) Einführung und Überblick. In: Automotive Software Engineering. ATZ/MTZ-Fachbuch. Springer Vieweg, Wiesbaden

IEC 61508 - SCOPUS

SC003 Blache, G. Handling index-out-of-bounds in safety-critical embedded C code using model-based development (2019) Software and Systems Modeling, 18 (3), pp. 1795-1807.

SC005 Aigner, A., Khelil, A. Assessment of model-based methodologies to architect cyber-physical systems (2019) ACM International Conference Proceeding Series, Part F148162, pp. 146-151.

SC006 Hochstrasser, M., Myschik, S., Holzapfel, F. Application of a process-oriented build tool for flight controller development along a DO-178C/DO-331 process (2019) Communications in Computer and Information Science, 991, pp. 380-405.

SC008 Allouch, A., Koubaa, A., Khalgui, M., Abbes, T. Qualitative and Quantitative Risk Analysis and Safety Assessment of Unmanned Aerial Vehicles Missions over the Internet (2019) IEEE Access, 7, art. no. 8695695, pp. 53392-53410.

SC011 Bussenot, R., Leblanc, H., Percebois, C. Orchestration of domain specific test languages with a behavior driven development approach (2018) 2018 13th System of Systems Engineering Conference, SoSE 2018, art. no. 8428788, pp. 431-437.

SC012 Getir, S., Grunske, L., Hoorn, A.V., Kehrer, T., Noller, Y., Tichy, M. Supporting semi-automatic co-evolution of architecture and fault tree models (2018) Journal of Systems and Software, 142, pp. 115-135.

SC013 Maro, S., Steghöfer, J.-P., Staron, M. Software traceability in the automotive domain: Challenges and solutions (2018) Journal of Systems and Software, 141, pp. 85-110.

SC017 Mhenni, F., Nguyen, N., Choley, J.-Y. SafeSysE: A Safety Analysis Integration in Systems Engineering Approach (2018) IEEE Systems Journal, 12 (1), pp. 161-172.

SC018 Kan, S., Huang, Z. Detecting safety-related components in statecharts through traceability and model slicing (2018) Software - Practice and Experience, 48 (3), pp. 428-448.

SC021 Hochstrasser, M., Myschik, S., Holzapfel, F. A process-oriented build tool for safety-critical model-based software development (2018) MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, 2018-January, pp. 191-202.

SC022 De La Vara, J.L., Ruiz, A., Espinoza, H. Recent advances towards the industrial application of model-driven engineering for assurance of safety-critical systems (2018) MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, 2018-January, pp. 632-641.

SC025 Sollfrank, M., Pirehgalin, M.F., Vogel-Heuser, B. Integration of safety aspects in modeling of Networked Control Systems (2017) Proceedings - 2017 IEEE 15th International Conference on Industrial Informatics, INDIN 2017, art. no. 8104806, pp. 405-412.

SC026 Barner, S., Diewald, A., Migge, J., Syed, A., Fohler, G., Faugere, M., Perez, D.G. DREAMS Toolchain: Model-Driven Engineering of Mixed-Criticality Systems (2017) Proceedings - ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, art. no. 8101272, pp. 259-269.

SC028 Brunner, M., Huber, M., Sauerwein, C., Breu, R. Towards an Integrated Model for Safety and Security Requirements of Cyber-Physical Systems (2017) Proceedings - 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017, art. no. 8004340, pp. 334-340.

SC030 Houliotis, K., Oikonomidis, P., Charchalakis, P., Stipidis, E. An efficient approach to designing mission-critical systems: Case study: Defensive Aid Suite (DAS) systems (2017) ICMT 2017 - 6th International Conference on Military Technologies, art. no. 7988793, pp. 402-409.

SC034 Iosif-Lazăr, A.F., Wąsowski, A. Trustworthy variant derivation with translation validation for safety critical product lines (2016) Journal of Logical and Algebraic Methods in Programming, 85 (6), pp. 1154-1176.

SC036 Aziz, M.W., Rashid, M. Domain specific modeling language for cyber physical systems (2016) Proceedings - 2016 International Conference on Information Systems Engineering, ICISE 2016, art. no. 7486209, pp. 29-33.

SC037 De La Vara, J.L., Ruiz, A., Attwood, K., Espinoza, H., Panesar-Walawege, R.K., López, Á., Del Río, I., Kelly, T. Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel (2016) Information and Software Technology, 72, pp. 16-30.

SC038 Kim, J.H., Kang, I., Kang, S., Boudjadar, A. A Process Algebraic Approach to Resource-Parameterized Timing Analysis of Automotive Software Architectures (2016) IEEE Transactions on Industrial Informatics, 12 (2), art. no. 7403990, pp. 655-671.

SC039 Lawford, M. Stupid tool tricks for smart model based design (2016) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9971 LNCS, pp. 1-7.

SC040 Zeller, M., Ratiu, D., Höfig, K. Towards the adoption of model-based engineering for the development of safety-critical systems in industrial practice (2016) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9923 LNCS, pp. 322-333.

SC043 Koark, F.J.U., Beul, C. Benefits of Functional Safety Re-Engineering (2015) SAE Technical Papers, 2015-April.

SC047 Folkesson, P., Ayatolahi, F., Sangchoolie, B., Vinter, J., Islam, M., Karlsson, J. Back-to-back fault injection testing in model-based development (2015) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9337, pp. 135-148.

SC051 Rupanov, V., Buckl, C., Fiege, L., Armbruster, M., Knoll, A., Spiegelberg, G. Employing early model-based safety evaluation to iteratively derive E/E architecture design (2014) Science of Computer Programming, 90 (PART B), pp. 161-179.

SC052 Preschern, C., Kajtazovic, N., Höller, A., Kreiner, C. Pattern-based safety development methods: Overview and comparison (2014) ACM International Conference Proceeding Series, 09-13-July-2014, art. no. 2721958.

SC054 Mayr, A., Plösch, R., Saft, M. Objective safety compliance checks for source code (2014) 36th International Conference on Software Engineering, ICSE Companion 2014 - Proceedings, pp. 115-124.

SC055 Dibbern, C., Hahn, A., Schweigert, S. Interoperability in co-simulatons of maritime systems (2014) Proceedings - 28th European Conference on Modelling and Simulation, ECMS 2014, pp. 71-77.

SC056 Falcone, A., Garro, A., Tundis, A. Modeling and simulation for the performance evaluation of the on-board communication system of a metro train (2014) 13th International Conference on Modeling and Applied Simulation, MAS 2014, pp. 20-29.

SC060 Mayr, A., Plosch, R., Saft, M. Objective measurement of safety in the context of IEC 61508-3 (2013) Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013, art. no. 6619487, pp. 45-52.

SC061 Sinha, R., Roop, P., Ranjitkar, P. Virtual traffic lights+ (2013) Transportation Research Record, (2381), pp. 73-80.

SC063 Prokhorova, Y., Troubitsyna, E. A survey of safety-oriented model-driven and formal development approaches (2013) International Journal of Critical Computer-Based Systems, 4 (2), pp. 93-118.

SC065 Sojer, D., Reichenbach, F., Ellevseth, S.-E., Buckl, C., Knoll, A. A model-driven approach for runtime reliability analysis (2013) Proceedings - 6th Latin-American Symposium on Dependable Computing, LADC 2013, art. no. 6542602, pp. 21-30.

SC066 Lano, K., Kolahdouz-Rahimi, S. High-integrity model- based development (2013) Progressions and Innovations in Model-Driven Software Engineering, pp. 1-17.

SC067 Singh, N.K. Using event-B for critical device software systems (2013) Using Event-B for Critical Device Software Systems, 9781447152606, pp. 1-326.

SC068 Panesar-Walawege, R.K., Sabetzadeh, M., Briand, L. Supporting the verification of compliance to safety standards via model-driven engineering: Approach, tool-support and empirical validation (2013) Information and Software Technology, 55 (5), pp. 836-864.

SC070 Fechtelkotter, P., Bleakley, G., Douglass, B.P., Amaba, B. Model-driven development for safety-critical projects in intelligent energy (2013) Society of Petroleum Engineers - SPE Middle East Intelligent Energy Conference and Exhibition 2013.

SC072 Garro, A., Tundis, A., Buffoni-Rogovchenko, L., Fritzson, P. From safety requirements to simulation-driven design of safe systems (2013) 12th International Conference on Modeling and Applied Simulation, MAS 2013, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2013, pp. 40-49.

SC073 Wasilewski, M., Hasselbring, W., Nowotka, D. Defining requirements on domain-specific languages in model-driven software engineering of safety-critical systems (2013) Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), P-215, pp. 467-482.

SC075 Turban, B. Tool-based requirement traceability between requirement and design artifacts (2013) Springer 4, 9783834824745, pp. 1-439.

SC076 Güdemann, M., Lipaczewski, M., Struck, S., Ortmeier, F. Unifying probabilistic and traditional formal model based analysis (2012) Tagungsband - Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VIII, MBEES 2012, pp. 123-132.

SC077 Hanai, R., Saito, H., Nakabo, Y., Fujiwara, K., Ogure, T., Mizuguchi, D., Homma, K., Ohba, K. RT-component based integration for IEC61508 ready system using SysML and IEC61499 function blocks (2012) 2012 IEEE/SICE International Symposium on System Integration, SII 2012, art. no. 6426952, pp. 105-110.

SC078 Lipaczewski, M., Struck, S., Ortmeier, F. Using tool-supported model based safety analysis - Progress and experiences in SAML development (2012) Proceedings of IEEE International Symposium on High Assurance Systems Engineering, art. no. 6375611, pp. 159-166.

SC081 Nejati, S., Sabetzadeh, M., Falessi, D., Briand, L., Coq, T. A SysML-based approach to traceability management and design slicing in support of safety certification: Framework, tool support, and case studies (2012) Information and Software Technology, 54 (6), pp. 569-590.

SC082 Hillenbrand, M., Heinz, M., Matheis, J., Müller-Glaser, K.D. Development of electric/electronic architectures for safety-related vehicle functions (2012) Software - Practice and Experience, 42 (7), pp. 817-851.

SC083 Panesar-Walawege, R.K., Sabetzadeh, M., Briand, L. Using model-driven engineering for managing safety evidence: Challenges, vision and experience (2011) Proceedings - 2011 1st International Workshop on Software Certification, WoSoCER 2011 - In Conjunction with the 22nd International Symposium on Software Reliability Engineering, ISSRE 2011, art. no. 6118521, pp. 7-12.

SC089 Sojer, D., Knoll, A., Buckl, C. Synthesis of diagnostic techniques based on an IEC 61508-aware metamodel (2011) SIES 2011 - 6th IEEE International Symposium on Industrial Embedded Systems, Conference Proceedings, art. no. 5953680, pp. 59-62.

SC091 Buckl, C., Sojer, D., Knoll, A. FTOS: Model-driven development of fault-tolerant automation systems (2010) Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2010, art. no. 5641211.

SC094 Conrad, M., Munier, P., Rauch, F. Qualifying software tools according to ISO 26262 (2010) Tagungsband - Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VI, MBEES 2010, pp. 117-128.

SC096 Snooke, N. Lessons from engineering: Can software benefit from product based evidence of reliability? (2010) ICSOFT 2010 - Proceedings of the 5th International Conference on Software and Data Technologies, 2, pp. 238-244.

SC097 Farkas, T., Klein, T., Röbig, H. Application of quality standards to multiple artifacts with a universal compliance solution (2010) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6100 LNCS, pp. 377-384.

SC098 Sedacca, B. Trust, but verify (2010) Engineering and Technology, 5 (13), pp. 32-35.

SC101 Conrad, M. Testing-based translation validation of generated code in the context of IEC 61508 (2009) Formal Methods in System Design, 35 (3), pp. 389-401.

SC102 Zander, J., Schieferdecker, I. Model-based testing of embedded systems exemplified for the automotive domain (2009) Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation, pp. 377-411.

SC105 Farkas, T. Quality improvement in automotive software engineering using a model-based approach (2008) Model-Driven Software Development: Integrating Quality Assurance, pp. 374-399.

IEC 61508 - Miscellaneous

D001 Mayr, Alois & Plösch, Reinhold & Saft, Matthias. (2011). Towards an Operational Safety Standard for Software: Modelling IEC 61508 Part 3. 97 - 104. 10.1109/ECBS.2011.8.

D002 Panesar-Walawege, Rajwinder & Sabetzadeh, Mehrdad & Briand, Lionel. (2011). A Model-Driven Engineering Approach to Support the Verification of Compliance to Safety Standards. Proceedings - International Symposium on Software Reliability Engineering, ISSRE. 30-39. 10.1109/ISSRE.2011.11.

D004 Luo Y., van den Brand M., Engelen L., Klabbers M. (2015) A Modeling Approach to Support Safety Assurance in the Automotive Domain. In: Selvaraj H., Zydek D., Chmaj G. (eds) Progress in Systems Engineering. Advances in Intelligent Systems and Computing, vol 366. Springer, Cham.

D005 Panesar-Walawege, Rajwinder & Sabetzadeh, Mehrdad & Briand, Lionel. (2011). Using UML Profiles for Sector-Specific Tailoring of Safety Evidence Information. 6998. 362-378.

ISO 26262 - SMS Publications

ISO 26262 - Springer

SP001 Rumpe B. (2017) Summary, Further Reading and Outlook. In: Agile Modeling with UML. Springer, Cham

SP004 Sporer H., Macher G., Höller A., Kreiner C. (2015) Bidirectional Crosslinking of System and Software Modeling in the Automotive Domain. In: Fantechi A., Pelliccione P. (eds) Software Engineering for Resilient Systems. SERENE 2015. Lecture Notes in Computer Science, vol 9274. Springer, Cham

SP006 Macher G., Armengaud E., Kreiner C. (2015) Integration of Heterogeneous Tools to a Seamless Automotive Toolchain. In: O'Connor R., Umay Akkaya M., Kemaneci K., Yilmaz M., Poth A., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2015. Communications in Computer and Information Science, vol 543. Springer, Cham

SP008 Schneider O., Mindel T., Liebmann J., Gonzalez Ramos R. (2015) Model based software development – solutions for series software. In: Bargende M., Reuss HC., Wiedemann J. (eds) 15. Internationales Stuttgarter Symposium. Proceedings. Springer Vieweg, Wiesbaden

SP009 Bertolino, A., Calabro', A., Di Giandomenico, F. et al. A tour of secure software engineering solutions for connected vehicles. Software Qual J 26, 1223–1256 (2018). https://doi.org/10.1007/s11219-017-9393-3

SP010 Sporer H. (2015) A Lean Automotive E/E-System Design Approach with Open Toolbox Access. In: O'Connor R., Umay Akkaya M., Kemaneci K., Yilmaz M., Poth A., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2015. Communications in Computer and Information Science, vol 543. Springer, Cham

SP011 Hamid, B., Gürgens, S. & Fuchs, A. Security patterns modeling and formalization for pattern-based development of secure software systems. Innovations Syst Softw Eng 12, 109–140 (2016). https://doi.org/10.1007/s11334-015-0259-1

SP012 Góngora H.G.C., Gaudré T., Tucci-Piergiovanni S. (2013) Towards an Architectural Design Framework for Automotive Systems Development. In: Aiguier M., Caseau Y., Krob D., Rauzy A. (eds) Complex Systems Design & Management. Springer, Berlin, Heidelberg

SP014 Martin H., Tschabuschnig K., Bridal O., Watzenig D. (2017) Functional Safety of Automated Driving Systems: Does ISO 26262 Meet the Challenges?. In: Watzenig D., Horn M. (eds) Automated Driving. Springer, Cham

SP015 Pohlmann, U., Hüwe, M. Model-driven allocation engineering: specifying and solving constraints based on the example of automotive systems. Autom Softw Eng 26, 315–378 (2019).
https://doi.org/10.1007/s10515-018-0248-3

SP016 Luo Y., Saberi A.K., den Brand M.. (2019) Safety-Driven Development and ISO 26262. In: Dajsuren Y., van den Brand M. (eds) Automotive Systems and Software Engineering. Springer, Cham

SP017 Luo Y., van den Brand M., Engelen L., Klabbers M. (2014) From Conceptual Models to Safety Assurance. In: Yu E., Dobbie G., Jarke M., Purao S. (eds) Conceptual Modeling. ER 2014. Lecture Notes in Computer Science, vol 8824. Springer, Cham

SP018 Staron M., Durisic D. (2017) AUTOSAR Standard. In: Automotive Software Architectures. Springer, Cham

SP020 Luo Y., van den Brand M., Engelen L., Favaro J., Klabbers M., Sartori G. (2013) Extracting Models from ISO 26262 for Reusable Safety Assurance. In: Favaro J., Morisio M. (eds) Safe and Secure Software Reuse. ICSR 2013. Lecture Notes in Computer Science, vol 7925. Springer, Berlin, Heidelberg

SP021 Adedjouma M., Dubois H., Terrier F., Kitouni T. (2012) Merging the Quality Assessment of Processes and Products in Automotive Domain. In: Dieste O., Jedlitschka A., Juristo N. (eds) Product-Focused Software Process Improvement. PROFES 2012. Lecture Notes in Computer Science, vol 7343. Springer, Berlin, Heidelberg

SP023 Mauborgne P., Deniaud S., Levrat E., Bonjour E., Micaëlli JP., Loise D. (2015) Preliminary Hazard Analysis Generation Integrated with Operational Architecture - Application to Automobile. In: Boulanger F., Krob D., Morel G., Roussel JC. (eds) Complex Systems Design & Management. Springer, Cham

SP024 Zhang R., Krishnan A. (2011) Using Delta Model for Collaborative Work of Industrial Large-Scaled E/E Architecture Models. In: Whittle J., Clark T., Kühne T. (eds) Model Driven Engineering Languages and Systems. MODELS 2011. Lecture Notes in Computer Science, vol 6981. Springer, Berlin, Heidelberg

SP025 Cichos H., Oster S., Lochau M., Schürr A. (2011) Model-Based Coverage-Driven Test Suite Generation for Software Product Lines. In: Whittle J., Clark T., Kühne T. (eds) Model Driven Engineering Languages and Systems. MODELS 2011. Lecture Notes in Computer Science, vol 6981. Springer, Berlin, Heidelberg

SP026 Luo Y., van den Brand M., Kiburse A. (2015) Safety Case Development with SBVR-Based Controlled Language. In: Desfray P., Filipe J., Hammoudi S., Pires L. (eds) Model-Driven Engineering and Software Development. MODELSWARD

2015. Communications in Computer and Information Science, vol 580. Springer, Cham

SP030 Gallina B., Szatmári Z. (2015) Ontology-Based Identification of Commonalities and Variabilities Among Safety Processes. In: Abrahamsson P., Corral L., Oivo M., Russo B. (eds) Product-Focused Software Process Improvement. PROFES 2015. Lecture Notes in Computer Science, vol 9459. Springer, Cham

SP033 Luo Y., Engelen L., van den Brand M. (2014) Metamodel Comparison and Model Comparison for Safety Assurance. In: Bondavalli A., Ceccarelli A., Ortmeier F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8696. Springer, Cham

SP034 Underwood S., Bartz D., Kade A., Crawford M. (2016) Truck Automation: Testing and Trusting the Virtual Driver. In: Meyer G., Beiker S. (eds) Road Vehicle Automation 3. Lecture Notes in Mobility. Springer, Cham

SP035 Domis D., Trapp M. (2008) Integrating Safety Analyses and Component-Based Design. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

SP036 Macher, G., Stolz, M., Armengaud, E. et al. Filling the gap between automotive systems, safety, and software engineering. Elektrotech. Inftech. 132, 142–148 (2015).
https://doi.org/10.1007/s00502-015-0301-x

SP037 Varró, D., Bergmann, G., Hegedüs, Á. et al. Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework. Softw Syst Model 15, 609–629 (2016).
https://doi.org/10.1007/s10270-016-0530-4

SP039 Staron M. (2019) Requirements Engineering for Automotive Embedded Systems. In: Dajsuren Y., van den Brand M. (eds) Automotive Systems and Software Engineering. Springer, Cham

SP041 Gonschorek T., Bergt P., Filax M., Ortmeier F. (2019) Integrating Safety Design Artifacts into System Development Models Using SafeDeML. In: Papadopoulos Y., Aslansefat K., Katsaros P., Bozzano M. (eds) Model-Based Safety and Assessment. IMBSA 2019. Lecture Notes in Computer Science, vol 11842. Springer, Cham

SP042 Beckers K., Côté I., Frese T., Hatebur D., Heisel M. (2015) A Structured Validation and Verification Method for Automotive Systems Considering the OEM/Supplier Interface. In: Koornneef F., van Gulijk C. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 9337. Springer, Cham

SP045 Mader R. et al. (2011) Computer-Aided PHA, FTA and FMEA for Automotive Embedded Systems. In: Flammini F., Bologna S., Vittorini V. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2011. Lecture Notes in Computer Science, vol 6894. Springer, Berlin, Heidelberg

SP048 Zoppelt M., Tavakoli Kolagari R. (2019) SAM: A Security Abstraction Model for Automotive Software Systems. In: Hamid B., Gallina B., Shabtai A., Elovici Y., Garcia-Alfaro J. (eds) Security and Safety Interplay of Intelligent Software Systems. CSITS 2018, ISSA 2018. Lecture Notes in Computer Science, vol 11552. Springer, Cham

SP050 Leitner, A., Mader, R., Kreiner, C. et al. A development methodology for variant-rich automotive software architectures. Elektrotech. Inftech. 128, 222–227 (2011). https://doi.org/10.1007/s00502-011-0001-0

SP051 Lantz J., Eliasson U. (2014) Scaling Agile Mechatronics: An Industrial Case Study. In: Bosch J. (eds) Continuous Software Engineering. Springer, Cham

SP052 Beckers K., Côté I., Frese T., Hatebur D., Heisel M. (2014) Systematic Derivation of Functional Safety Requirements for Automotive Systems. In: Bondavalli A., Di Giandomenico F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8666. Springer, Cham

SP054 Sojer, D., Buckl, C. & Knoll, A. Deriving fault-detection mechanisms from safety requirements. Comput Sci Res Dev 30, 21–34 (2015). https://doi.org/10.1007/s00450-011-0203-z

SP055 Fornari X. (2019) Battery Management System: From Safe Architecture Definition to System Simulation with Embedded Software. In: Langheim J. (eds) Electronic Components and Systems for Automotive Applications. Lecture Notes in Mobility. Springer, Cham

SP056 Chen D. et al. (2008) Modelling Support for Design of Safety-Critical Automotive Embedded Systems. In: Harrison M.D., Sujan MA. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2008. Lecture Notes in Computer Science, vol 5219. Springer, Berlin, Heidelberg

SP057 Fockel M. (2016) ASIL Tailoring on Functional Safety Requirements. In: Skavhaug A., Guiochet J., Schoitsch E., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2016. Lecture Notes in Computer Science, vol 9923. Springer, Cham

SP059 Moradi M., Van Acker B., Vanherpen K., Denil J. (2019) Model-Implemented Hybrid Fault Injection for Simulink (Tool Demonstrations). In: Chamberlain R., Taha W., Törngren M. (eds) Cyber Physical Systems. Model-Based Design. CyPhy 2018, WESE 2018. Lecture Notes in Computer Science, vol 11615. Springer, Cham

SP060 Dörr, H., Trögel, A. & Weinberg, D. Multiprojektfähigkeit in der Prüfung von Software-Modellen. ATZ Elektron 9, 52–55 (2014). https://doi.org/10.1365/s35658-014-0388-6

SP061 Adedjouma M., Dubois H., Terrier F., Kitouni T. (2012) An Experiment on Merging Quality Assessment in Automotive Domain. In: Mas A., Mesquida A., Rout T., O'Connor R.V., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2012. Communications in Computer and Information Science, vol 290. Springer, Berlin, Heidelberg

SP064 Zimmer B., Bürklen S., Knoop M., Höfflinger J., Trapp M. (2011) Vertical Safety Interfaces – Improving the Efficiency of Modular Certification. In: Flammini F., Bologna S., Vittorini V. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2011. Lecture Notes in Computer Science, vol 6894. Springer, Berlin, Heidelberg

SP066 Salay, R., Kokaly, S., Di Sandro, A. et al. Heterogeneous megamodel management using collection operators. Softw Syst Model 19, 231–260 (2020). https://doi.org/10.1007/s10270-019-00738-9

SP067 Mubeen, S., Nolte, T., Sjödin, M. et al. Supporting timing analysis of vehicular embedded systems through the refinement of timing constraints. Softw Syst Model 18, 39–69 (2019). https://doi.org/10.1007/s10270-017-0579-8

SP073 Wagner, M., Meroth, A. & Zöbel, D. Developing self-adaptive automotive systems. Des Autom Embed Syst 18, 199–221 (2014). https://doi.org/10.1007/s10617-013-9124-3

SP074 Chen, D., Johansson, R., Lönn, H. et al. Integrated safety and architecture modeling for automotive embedded systems*. Elektrotech. Inftech. 128, 196–202 (2011). https://doi.org/10.1007/s00502-011-0007-7

SP077 Khalil M. (2018) Pattern Libraries Guiding the Model-Based Reuse of Automotive Solutions. In: Khendek F., Gotzhein R. (eds) System Analysis and Modeling. Languages, Methods, and Tools for Systems Engineering. SAM 2018. Lecture Notes in Computer Science, vol 11150. Springer, Cham

SP078 Mubeen S., Bucaioni A. (2018) Modeling of Vehicular Distributed Embedded Systems: Transition from Single-Core to Multi-core. In: Latifi S. (eds) Information Technology - New Generations. Advances in Intelligent Systems and Computing, vol 558. Springer, Cham

SP080 Staron M. (2017) Software Architectures: Views and Documentation. In: Automotive Software Architectures. Springer, Cham

SP082 Castellanos Ardila J.P., Gallina B., Ul Muram F. (2018) Transforming SPEM 2.0-Compatible Process Models into Models Checkable for Compliance. In: Stamelos I., O'Connor R., Rout T., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2018. Communications in Computer and Information Science, vol 918. Springer, Cham

SP084 Staron M. (2017) Automotive Software Development. In: Automotive Software Architectures. Springer, Cham

SP085 Aubron M. (2015) KTM Functional Safety Environment – Break Silos, Ensure Full Traceability, Modularity and Automate Reporting. In: O'Connor R., Umay Akkaya M., Kemaneci K., Yilmaz M., Poth A., Messnarz R. (eds) Systems, Software and Services Process Improvement. EuroSPI 2015. Communications in Computer and Information Science, vol 543. Springer, Cham

SP086 Frese T., Gerber N., Hatebur D., Côté I., Heisel M. (2018) Functional Safety Processes and Advanced Driver Assistance Systems: Evolution or Revolution?. In: Proff H., Fojcik T. (eds) Mobilität und digitale Transformation. Springer Gabler, Wiesbaden

SP087 Broy, M. Mit Welcher Software Fährt das Auto der Zukunft?. ATZ Extra 16, 92–97 (2011).
https://doi.org/10.1365/s35778-011-0602-6

SP089 Kern, M., Wunder, M. & Stifter, C. Mehrganggetriebe mit Klauenkupplung für Hybridantriebe. ATZ Automobiltech Z 116, 44–47 (2014).
https://doi.org/10.1007/s35148-014-0404-4

SP090 Ginal, P., Dörr, H. & Lackner, H. Testautomatisierung bei Valeo Siemens eAutomotive. ATZ Elektron 12, 42–47 (2017).
https://doi.org/10.1007/s35658-017-0072-8

SP094 Gonschorek T., Zeller M., Höfig K., Ortmeier F. (2018) Fault Trees vs. Component Fault Trees: An Empirical Study. In: Gallina B., Skavhaug A., Schoitsch E., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2018. Lecture Notes in Computer Science, vol 11094. Springer, Cham

SP095 Adler N., Graf P., Müller-Glaser K.D. (2012) Model-Based Consistency Checks of Electric and Electronic Architectures against Requirements. In: Kienzle J. (eds) Models in Software Engineering. MODELS 2011. Lecture Notes in Computer Science, vol 7167. Springer, Berlin, Heidelberg

SP096 Sovani, S. Simulation beschleunigt Entwicklung des autonomen Fahrens. ATZ Automobiltech Z 119, 26–33 (2017).
https://doi.org/10.1007/s35148-017-0092-y

SP099 Frtunikj J., Rupanov V., Armbruster M., Knoll A. (2014) Adaptive Error and Sensor Management for Autonomous Vehicles: Model-Based Approach and Run-Time System. In: Ortmeier F., Rauzy A. (eds) Model-Based Safety and Assessment. IMBSA 2014. Lecture Notes in Computer Science, vol 8822. Springer, Cham

SP100 Wolf F. (2018) Softwareentwicklung in der Automobilindustrie. In: Fahrzeuginformatik. ATZ/MTZ-Fachbuch. Springer Vieweg, Wiesbaden

SP101 Saraoğlu M., Morozov A., Janschek K. (2019) Safety assessment of autonomous and connected vehicles by a model-based traffic simulation framework. In: Bargende M., Reuss HC., Wagner A., Wiedemann J. (eds) 19. Internationales Stuttgarter Symposium. Proceedings. Springer Vieweg, Wiesbaden

SP107 Naunheimer H., Bertsche B., Ryborz J., Novak W., Fietkau P. (2019) Getriebesteuerung – Elektrik, Elektronik, Aktuatorik und Sensorik. In: Fahrzeuggetriebe. Springer Vieweg, Berlin, Heidelberg

ISO 26262 - Scopus

SC001 Daun, M., Weyer, T., Pohl, K. Improving manual reviews in function-centered engineering of embedded systems using a dedicated review model (2019) Software and Systems Modeling, 18 (6), pp. 3421-3459.

SC004 Voelter, M., Kolb, B., Birken, K., Tomassetti, F., Alff, P., Wiart, L., Wortmann, A., Nordmann, A. Using language workbenches and domain-specific languages for safety-critical software development (2019) Software and Systems Modeling, 18 (4), pp. 2507-2530.

SC008 Krook, J., Svensson, L., Li, Y., Feng, L., Fabian, M. Design and formal verification of a safe stop supervisor for an automated vehicle (2019) Proceedings - IEEE International Conference on Robotics and Automation, 2019-May, art. no. 8793636, pp. 5607-5613.

SC013 Pietrusewicz, K. Metamodelling for design of mechatronic and cyber-physical systems (2019) Applied Sciences (Switzerland), 9 (3), art. no. 376.

SC014 Gonschorek, T., Bergt, P., Filax, M., Ortmeier, F., von Hoyningen-Hüne, J., Piper, T. SafeDeML: On integrating the safety design into the system model (2019) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11698 LNCS, pp. 271-285.

SC016 Heisig, P., Brink, C., Steghöfer, J.-P., Sachweh, S. A generic traceability metamodel for enabling unified end-to-end traceability in software product lines (2019) Proceedings of the ACM Symposium on Applied Computing, Part F147772, pp. 2344-2353.

SC018 Capilla, R., Gallina, B., Cetina, C., Favaro, J. Opportunities for software reuse in an uncertain world: From past to emerging trends (2019) Journal of Software: Evolution and Process, 31 (8), art. no. e2217.

SC019 Chechik, M., Salay, R., Viger, T., Kokaly, S., Rahimi, M. Software assurance in an uncertain world (2019) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11424 LNCS, pp. 3-21.

SC021 Amalfitano, D., De Simone, V., Fasolino, A.R. Exploiting ALM and MDE for supporting questionnaire-based gap analysis processes (2018) Proceedings - 44th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2018, art. no. 8498157, pp. 1-8.

SC023 Getir, S., Grunske, L., Hoorn, A.V., Kehrer, T., Noller, Y., Tichy, M. Supporting semi-automatic co-evolution of architecture and fault tree models (2018) Journal of Systems and Software, 142, pp. 115-135.

SC026 Maro, S., Steghöfer, J.-P., Staron, M. Software traceability in the automotive domain: Challenges and solutions (2018) Journal of Systems and Software, 141, pp. 85-110.

SC027 Munk, P., Abele, A., Thaden, E., Nordmann, A., Amarnath, R., Schweizer, M., Burton, S. INVITED: Semi-automatic safety analysis and optimization (2018) Proceedings - Design Automation Conference, Part F137710, art. no. a29.

SC028 Gheraibia, Y., Kabir, S., Djafri, K., Krimou, H. An Overview of the Approaches for Automotive Safety Integrity Levels Allocation (2018) Journal of Failure Analysis and Prevention, 18 (3), pp. 707-720.

SC030 Voss, S., Eder, J. Handling system complexity in sCPS: Usable design space exploration (2018) Proceedings - International Conference on Software Engineering, pp. 2-5.

SC033 Mhenni, F., Nguyen, N., Choley, J.-Y. SafeSysE: A Safety Analysis Integration in Systems Engineering Approach (2018) IEEE Systems Journal, 12 (1), pp. 161-172.

SC034 Korssen, T., Dolk, V., Van De Mortel-Fronczak, J., Reniers, M., Heemels, M. Systematic Model-Based Design and Implementation of Supervisors for Advanced Driver Assistance Systems (2018) IEEE Transactions on Intelligent Transportation Systems, 19 (2), art. no. 8226993, pp. 533-544.

SC035 Damak, Y., Jankovic, M., Leroy, Y., Yannou, B. Analysis of safety requirements evolution in the transition of land transportation systems toward autonomy (2018) Proceedings of International Design Conference, DESIGN, 6, pp. 2845-2854.

SC038 Fung, N.L.S., Kokaly, S., Di Sandro, A., Salay, R., Chechik, M. MMINT-A: A tool for automated change impact assessment on assurance cases (2018) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11094 LNCS, pp. 60-70.

SC040 De La Vara, J.L., Ruiz, A., Espinoza, H. Recent advances towards the industrial application of model-driven engineering for assurance of safety-critical systems (2018) MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, 2018-January, pp. 632-641.

SC045 Törngren, M., Sellgren, U. Complexity Challenges in Development of Cyber-Physical Systems (2018) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10760 LNCS, pp. 478-503.

SC046 Milanovic, M., Rodic, M., Truntic, M. Functional safety in power electronics converters (2017) International Conference on Electical Drives and Power Electronics, 2017-October, pp. 1-14.

SC047 Bertram, V., Maoz, S., Ringert, J.O., Rumpe, B., Von Wenckstern, M. Component and Connector Views in Practice: An Experience Report (2017) Proceedings - ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, art. no. 8101260, pp. 167-177.

SC048 Eder, J., Zverlov, S., Voss, S., Khalil, M., Ipatiov, A. Bringing DSE to Life: Exploring the Design Space of an Industrial Automotive Use Case (2017) Proceedings - ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems, MODELS 2017, art. no. 8101273, pp. 270-280.

SC051 Brunner, M., Huber, M., Sauerwein, C., Breu, R. Towards an Integrated Model for Safety and Security Requirements of Cyber-Physical Systems (2017) Proceedings - 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017, art. no. 8004340, pp. 334-340.

SC054 Kokaly, S. Managing assurance cases in model based software systems (2017) Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017, art. no. 7965382, pp. 453-456.

SC055 Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach (2017) Proceedings - 2017 IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice, SER and IP 2017, art. no. 7964358, pp. 3-9.

SC057 Bucher, H., Reichmann, C., Becker, J. An Integrated Approach Enabling Cross-Domain Simulation of Model-Based E/E-Architectures (2017) SAE Technical Papers, 2017-March (March).

SC059 Kokaly, S., Salay, R., Chechik, M., Lawford, M., Maibaum, T. Safety case impact assessment in automotive software systems: An improved model-based approach (2017) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10488 LNCS, pp. 69-85.

SC062 Grazioli, F., Kusmenko, E., Roth, A., Rumpe, B., Von Wenckstern, M. Simulation framework for executing component and connector models of self-driving vehicles (2017) CEUR Workshop Proceedings, 2019, pp. 109-115.

SC065 Maro, S., Staron, M., Steghöfer, J.-P. Challenges of establishing traceability in the automotive domain (2017) Lecture Notes in Business Information Processing, 269, pp. 153-172.

SC066 Amarnath, R., Munk, P., Thaden, E., Nordmann, A., Burton, S. Dependability Challenges in the Model-Driven Engineering of Automotive Systems (2016) Proceedings - 2016 IEEE 27th International Symposium on Software Reliability Engineering Workshops, ISSREW 2016, art. no. 7789365, pp. 1-4.

SC069 Blache, G. Handling index-out-of-bounds in safety-critical embedded C code using model-based development (2016) Proceedings - 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2016, pp. 143-149.

SC070 Kokaly, S., Salay, R., Cassano, V., Maibaum, T., Chechik, M. A model management approach for assurance case reuse due to system evolution (2016) Proceedings - 19th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2016, pp. 196-206.

SC071 Holtmann, J., Bernijazov, R., Meyer, M., Schmelter, D., Tschirner, C. Integrated and iterative systems engineering and software requirements engineering for technical systems (2016) Journal of Software: Evolution and Process, 28 (9), pp. 722-743.

SC072 Mauborgne, P., Deniaud, S., Levrat, E., Bonjour, E., Micaëlli, J.-P., Loise, D. Operational and System Hazard Analysis in a Safe Systems Requirement Engineering Process - Application to automotive industry (2016) Safety Science, 87, pp. 256-268.

SC074 Gerlitz, T., Kowalewski, S. Architectural analysis of MATLAB/Simulink models with Artshop (2016) Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016, art. no. 7516853, pp. 307-310.

SC076 De La Vara, J.L., Ruiz, A., Attwood, K., Espinoza, H., Panesar-Walawege, R.K., López, Á., Del Río, I., Kelly, T. Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel (2016) Information and Software Technology, 72, pp. 16-30.

SC078 Shahrokni, A., Gergely, P., Söderberg, J., Pelliccione, P. Organic Evolution of Development Organizations - An Experience Report (2016) SAE Technical Papers.

SC079 Luo, Y., Li, Z., Van Den Brand, M. A categorization of GSN-based safety cases and patterns (2016) MODELSWARD 2016 - 4th International Conference on Model-Driven Engineering and Software Development, Doctoral Consortium, 2016-February, pp. 509-516.

SC082 Salay, R., Kokaly, S., Chechik, M., Maibaum, T. Heterogeneous megamodel slicing for model evolution (2016) CEUR Workshop Proceedings, 1706, pp. 50-59.

SC083 Behere, S., Törngren, M. Systems engineering and architecting for intelligent autonomous systems (2016) Automated Driving: Safer and More Efficient Future Driving, pp. 313-351.

SC084 Wägemann, T., Langer, T., Mottok, J., Osinski, L., Stappert, F., Kolagari, R.T. Models for dependable heterogenous multi- and many-core system software design revisited (2016) ARCS 2016 - 29th International Conference on Architecture of Computing Systems, 2016-April, pp. 1-8.

SC085 Papadopoulos, Y., Walker, M., Parker, D., Sharvia, S., Bottaci, L., Kabir, S., Azevedo, L., Sorokos, I. A synthesis of logic and bio-inspired techniques in the design of dependable systems (2016) Annual Reviews in Control, 41, pp. 170-182.

SC086 Meroth, A.M., Tränkle, F., Richter, B.F., Wagner, M., Neher, M., Lüling, J. Functional Safety and Development Process Capability for Intelligent Transportation Systems (2015) IEEE Intelligent Transportation Systems Magazine, 7 (4), art. no. 7302638, pp. 12-23.

SC088 Sporer, H. A model-based domain-specific language approach for the automotive E/E-System design (2015) Proceeding of the 2015 Research in Adaptive and Convergent Systems, RACS 2015, pp. 357-362.

SC089 Macher, G., Armengaud, E., Obendrauf, R., Kreiner, C. Automated generation of basic software configuration of embedded systems (2015) Proceeding of the 2015 Research in Adaptive and Convergent Systems, RACS 2015, pp. 461-464.

SC091 Klas, M., Bauer, T., Dereani, A., Soderqvist, T., Helle, P. A Large-Scale Technology Evaluation Study: Effects of Model-based Analysis and Testing (2015) Proceedings - International Conference on Software Engineering, 2, art. no. 7202956, pp. 119-128.

SC092 Kaiser, C., Herbst, B. Smart engineering for smart factories: How OSLC could enable plug & play tool integration (2015) Mensch und Computer 2015 - Workshop, pp. 269-280.

SC093 Macher, G., Stolz, M., Armengaud, E., Kreiner, C. Filling the gap between automotive systems, safety, and software engineering [Integration von System-, Safety- und Software-Entwicklung im automobilen Umfeld] (2015) Elektrotechnik und Informationstechnik, 132 (3), pp. 142-148.

SC094 Koark, F.J.U., Beul, C. Benefits of Functional Safety Re-Engineering (2015) SAE Technical Papers, 2015-April (April).

SC096 Mauborgne, P., Deniaud, S., Levrat, E., Bonjour, E., Micaëlli, J.-P., Loise, D. Preliminary hazard analysis generation integrated with operational architecture-application to automobile (2015) Complex Systems Design and Management - Proceedings of the 5th International Conference on Complex Systems Design and Management, CSD and M 2014, pp. 297-309.

SC100 Baumgart, S., Fröberg, J., Punnekkat, S. Enhancing model-based engineering of product lines by adding functional safety (2015) CEUR Workshop Proceedings, 1487, pp. 53-62.

SC103 De Oliveira, A.L., Braga, R.T.V., Masiero, P.C., Papadopoulos, Y., Habli, I., Kelly, T. A Model-Based Approach to Support the Automatic Safety Analysis of Multiple Product Line Products (2015) Brazilian Symposium on Computing System Engineering, SBESC, 2015-April, art. no. 7091158, pp. 7-12.

SC106 Folkesson, P., Ayatolahi, F., Sangchoolie, B., Vinter, J., Islam, M., Karlsson, J. Back-to-back fault injection testing in model-based development (2015) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9337, pp. 135-148.

SC107 Bonfiglio, V., Montecchi, L., Rossi, F., Lollini, P., Pataricza, A., Bondavalli, A. Executable Models to Support Automated Software FMEA (2015) Proceedings of IEEE International Symposium on High Assurance Systems Engineering, 2015-January (January), art. no. 7027431, pp. 189-196.

SC108 Hawkins, R., Habli, I., Kolovos, D., Paige, R., Kelly, T. Weaving an Assurance Case from Design: A Model-Based Approach (2015) Proceedings of IEEE International Symposium on High Assurance Systems Engineering, 2015-January (January), art. no. 7027421, pp. 110-117.

SC110 Hodlen, T., Dickerson, C., Luff, R. Preliminary report: A proposed model based systems engineering approach to a virtual vehicle architecture model (V2AM) for live-virtual testing and Prototyping (2014) Proceedings - IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2014, art. no. 6899174, pp. 384-390.

SC111 Rupanov, V., Buckl, C., Fiege, L., Armbruster, M., Knoll, A., Spiegelberg, G. Employing early model-based safety evaluation to iteratively derive E/E architecture design (2014) Science of Computer Programming, 90 (PART B), pp. 161-179.

SC112 Lovric, T., Schneider-Scheyer, M., Sarkic, S. SysML as backbone for engineering and safety - Practical experience with TRW braking ECU (2014) SAE Technical Papers, 1.

SC113 Dariusz, S., Bert, D., Yoann, D., Marc, V.V. Model-based and scalable functional safety engineering methodology for on-and off-highway vehicles (2014) FISITA 2014 World Automotive Congress - Proceedings, .

SC117 Groza, A., Marc, N. Consistency checking of safety arguments in the Goal Structuring Notation standard (2014) Proceedings - 2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing, ICCP 2014, art. no. 6936981, pp. 59-66.

SC121 Macher, G., Armengaud, E., Kreiner, C. Automated generation of AUTOSAR description file for safety-critical software architectures (2014) Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), P-232, pp. 2145-2156.

SC123 Mader, R., Obendrauf, R., Prinz, P., Griessnig, G. Experience report: A safety engineering tool supporting error model creation and visualization (2014) Proceedings - International Symposium on Software Reliability Engineering, ISSRE, art. no. 6982632, pp. 255-266.

SC126 Oertel, M., Kacimi, O., Böde, E. Proving compliance of implementation models to safety specifications (2014) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8696 LNCS, pp. 97-107.

SC127 Baumgart, A., Ellen, C. A recipe for tool interoperability (2014) MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development, pp. 300-308.

SC131 Chen, Y.-Y., Peng, J.-X. Development of safety process in model-based design platform for safety-critical systems (2013) Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, art. no. 6615386, pp. 627-630.

SC132 Fischer, K., Krumeich, J., Panfilenko, D., Born, M., Desfray, P. Viewpoint-based modeling: A stakeholder-centered approach for model-driven engineering (2013) Advances and Applications in Model-Driven Engineering, pp. 317-341.

SC136 Mader, R., Armengaud, E., Grießnig, G., Kreiner, C., Steger, C., Weiß, R. OASIS: An automotive analysis and safety engineering instrument (2013) Reliability Engineering and System Safety, 120, pp. 150-162.

SC141 Mader, R., Grießnig, G., Armengaud, E., Leitner, A., Kreiner, C., Bourrouilh, Q., Steger, C., Weiß, R. A bridge from system to software development for safety-critical automotive embedded systems (2012) Proceedings - 38th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2012, art. no. 6328131, pp. 75-79.

SC145 Fockel, M., Holtmann, J., Meyer, J. Semi-automatic establishment and maintenance of valid traceability in automotive development processes (2012) 2012 2nd International Workshop on Software Engineering for Embedded Systems, SEES 2012 - Proceedings, art. no. 6225489, pp. 37-43.

SC150 Hillenbrand, M., Heinz, M., Matheis, J., Müller-Glaser, K.D. Development of electric/electronic architectures for safety-related vehicle functions (2012) Software - Practice and Experience, 42 (7), pp. 817-851.

SC152 Mader, R., Grießnig, G., Leitner, A., Kreiner, C., Bourrouilh, Q., Armengaud, E., Steger, C., Weiß, R. A computer-aided approach to preliminary hazard analysis for automotive embedded systems (2011) Proceedings - 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, ECBS 2011, art. no. 5934817, pp. 169-178.

SC157 David, P., Shawky, M. Supporting ISO 26262 with SysML, benefits and limits (2010) Reliability, Risk and Safety: Back to the Future, pp. 2000-2008.

ISO 26262 - Miscellaneous

D001 Ross, H. (2013) Funktionale Sicherheit im Automobil. Carl Hanser Verlag, München (Germany).

D002 Herrmann, S., Duerholz, D., Staerk, R., Kriso, S. (2015) SAFETY Essentials: ISO 26262 at a glance. Kugler Maag Cie, Kornwestheim (Germany).

D003 Adler, N. (2014) Modellbasierte Entwicklung funktional sicherer Hardware nach ISO 26262. KIT Scientific Publishing, Karlsruhe (Germany), isbn: 978-3-7315-0442-9.

D005 Kannan, Manoj & Barosan, Ion & Dajsuren, Yanja & Luo, Yaping. (2015). Analysis of ISO 26262 Compliant Techniques for the Automotive Domain.

D006 Rana, R., Staron, M., Berger, C., Hansson, J., Nilsson, M., & Törner, F. (2013). Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model based Development. ICSOFT.

D007 Sporer, Harald & Macher, Georg & Armengaud, Eric & Kreiner, Christian. (2015). Incorporation of Model-Based System and Software Development Environments. 177-180. 10.1109/SEAA.2015.65.

D008 Gallina, Barbara. (2014). A Model-Driven Safety Certification Method for Process Compliance. Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014. 204-209. 10.1109/ISSREW.2014.30.

D009 Born, Marc & Favaro, John & Kath, Olaf. (2010). Application of ISO DIS 26262 in practice. 3-6. 10.1145/1772643.1772645.

D011 Hillenbrand, M. (2012) Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen. KIT Scientific Publishing, Karlsruhe (Germany), isbn: 978-3-86644-803-2.

D012 Burton, D.S., & Habermann, A. (2011). Automotive Systems Engineering und Functional Safety : The Way Forward.

D013 Alt, O. (2014) Survive ISO 26262 with model-based development!. embedded world conference 2014.

D015 Luo, Yaping & Brand, M. & Engelen, Luc & Klabbers, Martijn. (2015). A Modeling Approach to Support Safety Assurance in the Automotive Domain. Advances in Intelligent Systems and Computing. 1089. 339-345.

DO-178 - SMS Publications

DO-178 - Springer

SP006 Mkaouar, H., Zalila, B., Hugues, J. et al. A formal approach to AADL model-based software engineering. Int J Softw Tools Technol Transfer (2019). https://doi.org/10.1007/s10009-019-00513-7

SP009 Raistrick C., Bloomfield T. (2004) Model Driven Architecture – An Industry Perspective. In: de Lemos R., Gacek C., Romanovsky A. (eds) Architecting Dependable Systems II. Lecture Notes in Computer Science, vol 3069. Springer, Berlin, Heidelberg

SP010 Zoughbi, G., Briand, L. & Labiche, Y. Modeling safety and airworthiness (RTCA DO-178B) information: conceptual model and UML profile. Softw Syst Model 10, 337–367 (2011). https://doi.org/10.1007/s10270-010-0164-x

SP018 Zolotas, A., Hoyos Rodriguez, H., Hutchesson, S. et al. Bridging proprietary modelling and open-source model management tools: the case of PTC Integrity Modeller and Epsilon. Softw Syst Model 19, 17–38 (2020). https://doi.org/10.1007/s10270-019-00732-1

SP019 Bender O., Hiller M., Girod M., Strobel C., Waßmuth M., Dieudonné L. (2012) Application and Evaluation in the Avionics Domain. In: Pohl K., Hönninger H., Achatz R., Broy M. (eds) Model-Based Engineering of Embedded Systems. Springer, Berlin, Heidelberg

SP024 Balogh A. et al. (2010) Workflow-Driven Tool Integration Using Model Transformations. In: Engels G., Lewerentz C., Schäfer W., Schürr A., Westfechtel B. (eds) Graph Transformations and Model-Driven Engineering. Lecture Notes in Computer Science, vol 5765. Springer, Berlin, Heidelberg

SP026 Huhn M., Hungar H. (2010) 8 UML for Software Safety and Certification. In: Giese H., Karsai G., Lee E., Rumpe B., Schätz B. (eds) Model-Based Engineering of Embedded Real-Time Systems. MBEERTS 2007. Lecture Notes in Computer Science, vol 6100. Springer, Berlin, Heidelberg

SP038 Gallina B., Pitchai K.R., Lundqvist K. (2014) S-TunExSPEM: Towards an Extension of SPEM 2.0 to Model and Exchange Tunable Safety-Oriented Processes. In: Lee R. (eds) Software Engineering Research, Management and Applications. Studies in Computational Intelligence, vol 496. Springer, Heidelberg

SP039 Luo Y., van den Brand M., Kiburse A. (2015) Safety Case Development with SBVR-Based Controlled Language. In: Desfray P., Filipe J., Hammoudi S., Pires L. (eds) Model-Driven Engineering and Software Development. MODELSWARD

2015. Communications in Computer and Information Science, vol 580. Springer, Cham

SP040 Luo Y., Engelen L., van den Brand M. (2014) Metamodel Comparison and Model Comparison for Safety Assurance. In: Bondavalli A., Ceccarelli A., Ortmeier F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8696. Springer, Cham

SP053 Peleska J., Brauer J., Huang W. (2018) Model-Based Testing for Avionic Systems Proven Benefits and Further Challenges. In: Margaria T., Steffen B. (eds) Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice. ISoLA 2018. Lecture Notes in Computer Science, vol 11247. Springer, Cham

SP056 Kesserwan, N., Dssouli, R., Bentahar, J. et al. From use case maps to executable test procedures: a scenario-based approach. Softw Syst Model 18, 1543–1570 (2019).
https://doi.org/10.1007/s10270-017-0620-y

SP057 Hutchesson S., McDermid J. (2010) Development of High-Integrity Software Product Lines Using Model Transformation. In: Schoitsch E. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2010. Lecture Notes in Computer Science, vol 6351. Springer, Berlin, Heidelberg

SP062 de la Vara J.L., Marín B., Ayora C., Giachetti G. (2017) An Experimental Evaluation of the Understanding of Safety Compliance Needs with Models. In: Mayr H., Guizzardi G., Ma H., Pastor O. (eds) Conceptual Modeling. ER 2017. Lecture Notes in Computer Science, vol 10650. Springer, Cham

SP066 Sojer, D., Buckl, C. & Knoll, A. Deriving fault-detection mechanisms from safety requirements. Comput Sci Res Dev 30, 21–34 (2015).
https://doi.org/10.1007/s00450-011-0203-z

SP076 Hamid, I., Zalila, B., Najm, E. et al. Automatic framework generation for hard real-time applications. Innovations Syst Softw Eng 4, 107–122 (2008).
https://doi.org/10.1007/s11334-008-0044-5

SP079 Philip G., D'Souza M. (2018) Model-Based Safety Validation for Embedded Real-Time Systems. In: Nanda M., Jeppu Y. (eds) Formal Methods for Safety and Security. Springer, Singapore

DO-178 - Scopus

SC002 Schumann, J., Goseva-Popstojanova, K. Verification and validation approaches for model-based software engineering (2019) Proceedings - 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion, MODELS-C 2019, art. no. 8904785, pp. 514-518.

SC003 Iqbal, M.Z., Sartaj, H., Khan, M.U., Ul Haq, F., Qaisar, I. A Model-Based Testing Approach for Cockpit Display Systems of Avionics (2019) Proceedings - 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems, MODELS 2019, art. no. 8906961, pp. 67-77.

SC004 Paz, A., El Boussaidi, G. Supporting consistency in the heterogeneous design of safety-critical software (2019) Proceedings - International Computer Software and Applications Conference, 1, art. no. 8754292, pp. 37-46.

SC005 Paz, A., El Boussaidi, G. A requirements modelling language to facilitate avionics software verification and certification (2019) Proceedings - 2019 IEEE/ACM 6th International Workshop on Requirements Engineering and Testing, RET 2019, art. no. 8843241, pp. 1-8.

SC008 Li, L., Soskin, N.L., Jbara, A., Karpel, M., Dori, D. Model-Based Systems Engineering for Aircraft Design with Dynamic Landing Constraints Using Object-Process Methodology (2019) IEEE Access, 7, art. no. 8710233, pp. 61494-61511.

SC009 Hochstrasser, M., Myschik, S., Holzapfel, F. Application of a process-oriented build tool for flight controller development along a DO-178C/DO-331 process (2019) Communications in Computer and Information Science, 991, pp. 380-405.

SC021 Peleska, J. Model-based avionic systems testing for the airbus family (2018) Proceedings of the European Test Workshop, 2018-May, pp. 1-10.

SC022 Semeráth, O., Nagy, A.S., Varró, D. A graph solver for the automated generation of consistent domain-specific models (2018) Proceedings - International Conference on Software Engineering, pp. 969-980.

SC023 Jafer, S., Durak, U., Aydemir, H., Ruff, R., Pawletta, T. Advances in software engineering and aeronautics (2018) Advances in Aeronautical Informatics: Technologies Towards Flight 4.0, pp. 87-102.

SC026 Paz, A., Boussaidi, G.E. Building a software requirements specification and design for an avionics system: An experience report (2018) Proceedings of the ACM Symposium on Applied Computing, pp. 1262-1271.

SC029 Philip, G., Suresh, V.P., D'Souza, M. Safety validation using AADL system architecture models (2018) ACM International Conference Proceeding Series, art. no. a7.

SC031 Mathew, P.G., Liscouet-Hanke, S., Le Masson, Y. Model-Based Systems Engineering Methodology for Implementing Networked Aircraft Control System on Integrated Modular Avionics - Environmental Control System Case Study (2018) SAE Technical Papers, 2018-November.

SC032 Hochstrasser, M., Myschik, S., Holzapfel, F. A process-oriented build tool for safety-critical model-based software development (2018) MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, 2018-January, pp. 191-202.

SC036 Blooshi, M.A., Jafer, S., Patel, K. Review of formal agile methods as cost-effective airworthiness certification processes (2018) Journal of Aerospace Information Systems, 15 (8), pp. 471-484.

SC041 Golra, F.R., Dagnat, F., Souquières, J., Sayar, I., Guerin, S. Bridging the gap between informal requirements and formal specifications using model federation (2018) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10886 LNCS, pp. 54-69.

SC045 Milanovic, M., Rodic, M., Truntic, M. Functional safety in power electronics converters (2017) International Conference on Electical Drives and Power Electronics, 2017-October, pp. 1-14.

SC046 Hemmati, H., Arefin, S.S., Siddiqui, T.R. Analytics-based safety monitoring and verification (2017) 2017 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2017, 2017-January, pp. 3608-3613.

SC048 Philip, G., Dsouza, M., Abidha, V.P. Model based safety analysis: Automatic generation of safety validation test cases (2017) AIAA/IEEE Digital Avionics Systems Conference - Proceedings, 2017-September, art. no. 8102108.

SC051 Wouters, L., Creff, S., Bella, E.E., Koudri, A. Collaborative systems engineering: Issues & challenges (2017) Proceedings of the 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design, CSCWD 2017, art. no. 8066742, pp. 486-491.

SC057 Leonard, S., Olszewska, J.I. Model-based development of interactive multimedia system (2017) 2017 3rd IEEE International Conference on Cybernetics, CYBCONF 2017 - Proceedings, art. no. 7985791.

SC058 Semeráth, O., Barta, Á., Horváth, Á., Szatmári, Z., Varró, D. Formal validation of domain-specific languages with derived features and well-formedness constraints (2017) Software and Systems Modeling, 16 (2), pp. 357-392.

SC064 Paz, A., El Boussaidi, G. On the Exploration of Model-Based Support for DO-178C-Compliant Avionics Software Development and Certification (2016) Proceedings - 2016 IEEE 27th International Symposium on Software Reliability Engineering Workshops, ISSREW 2016, art. no. 7789405, pp. 229-236.

SC065 Gallina, B., Andrews, A. Deriving verification-related means of compliance for a model-based testing process (2016) AIAA/IEEE Digital Avionics Systems Conference - Proceedings, 2016-December, art. no. 7778046.

SC067 De La Vara, J.L., Marín, B., Giachetti, G., Ayora, C. Do Models Improve the Understanding of Safety Compliance Needs?: Insights from a Pilot Experiment (2016) International Symposium on Empirical Software Engineering and Measurement, 08-09-September-2016, art. no. a32.

SC071 Basagiannis, S. Software certification of airborne cyber-physical systems under DO-178C (2016) Proceedings of the 2016 Workshop on Symbolic and Numerical Methods for Reachability Analysis, SNR 2016 - Held as Part of CPS Week, art. no. 7479378.

SC072 De La Vara, J.L., Ruiz, A., Attwood, K., Espinoza, H., Panesar-Walawege, R.K., López, Á., Del Río, I., Kelly, T. Model-based specification of safety compliance needs for critical systems: A holistic generic metamodel (2016) Information and Software Technology, 72, pp. 16-30.

SC075 Simi, S.M., Mulholland, S.P., Merritt, L.B. Next-generation model-based systems engineering processes and tools supporting the airworthiness efforts of Cyber Physical Systems (CPS) (2016) Annual Forum Proceedings - AHS International, 4, pp. 3068-3075.

SC076 Allen, J.L. An overview of model-based development verification/validation processes and technologies in the aerospace industry (2016) AIAA Modeling and Simulation Technologies Conference, 12 p.

SC077 Lawford, M. Stupid tool tricks for smart model based design (2016) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9971 LNCS, pp. 1-7.

SC078 Alajrami, S., Gallina, B., Sljivo, I., Romanovsky, A., Isberg, P. Towards cloud-based enactment of safety-related processes (2016) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9922 LNCS, pp. 309-321.

SC079 Prosvirin, D.A., Kharchenko, V.P. Model-based solution and software engineering environment for UAV critical onboard applications (2015) 2015 IEEE 3rd International Conference Actual Problems of Unmanned Aerial Vehicles Developments, APUAVD 2015 - Proceedings, art. no. 7346629, pp. 312-315.

SC081 Klas, M., Bauer, T., Dereani, A., Soderqvist, T., Helle, P. A Large-Scale Technology Evaluation Study: Effects of Model-based Analysis and Testing (2015) Proceedings - International Conference on Software Engineering, 2, art. no. 7202956, pp. 119-128.

SC082 Gaeta, J.P., Czarnecki, K. Modeling aerospace systems product lines in SysML (2015) ACM International Conference Proceeding Series, 20-24-July-2015, pp. 293-302.

SC084 Grant, E.S., Datta, T. Roadmap to a DO-178C formal model-based software engineering methodology (2015) Lecture Notes in Engineering and Computer Science, 1, pp. 460-465.

SC086 Amundson, I., Shipton, L., Liu, A., Nowak, M. Toward efficient model-based development of aerospace applications (2015) 15th AIAA Aviation Technology, Integration, and Operations Conference, 6 p.

SC087 Lin, H., Wu, J., Yuan, C., Luo, Y., Van Den Brand, M., Engelen, L. A systematic approach for safety evidence collection in the safety-critical domain (2015) 9th Annual IEEE International Systems Conference, SysCon 2015 - Proceedings, art. no. 7116751, pp. 194-199.

SC088 Wu, J., Yue, T., Ali, S., Zhang, H. A modeling methodology to facilitate safety-oriented architecture design of industrial avionics software (2015) Software - Practice and Experience, 45 (7), pp. 893-924.

SC092 Lano, K., Kolahdouz-Rahimi, S. High-integrity model-based development (2014) Handbook of Research on Innovations in Systems and Software Engineering, pp. 479-499.

SC093 Simi, S.M., Mulholland, S.P., Tanner, W.G. TES-SAVi AWESUM^{TM} model-based systems engineering (MBSE) for FACETM applications (2014) IEEE Aerospace Conference Proceedings, art. no. 6836220.

SC094 Tavares, B., Cintra, J., Alves, R. IMADE: Integrated modular avionic development environment (2014) AIAA/IEEE Digital Avionics Systems Conference - Proceedings, art. no. 6979475, pp. 4D61-4D615.

SC097 Dierkes, M. Combining test and proof in MBAT: An aerospace case study (2014) MODELSWARD 2014 - Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development, pp. 636-644.

SC098 Luo, Y., Van Den Brand, M., Engelen, L., Klabbers, M. From conceptual models to safety assurance (2014) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8824, pp. 195-208.

SC107 De La Vara, J.L., Panesar-Walawege, R.K. SafetyMet: A metamodel for safety standards (2013) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8107 LNCS, pp. 69-86.

SC109 Bernardi, S., Merseguer, J., Petriu, D.C. Model-driven dependability assessment of software systems (2013) Model-Driven Dependability Assessment of Software Systems, 9783642395123, pp. 1-187.

SC113 Eriksson, A., Lindström, B., Offutt, J. Transformation rules for platform independent testing: An empirical study (2013) Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013, art. no. 6569732, pp. 202-211.

SC115 Nanda, M., Jayanthi, J., Jamadagni, C.S., Madhan, V. Quantitative metrics for improving software performance for an integrated tool platform (2013) SysCon 2013 - 7th Annual IEEE International Systems Conference, Proceedings, art. no. 6549930, pp. 512-519.

SC121 Insaurralde, C.C., Seminario, M.A., Jiménez, J.F., Giron-Sierra, J.M. Model-driven system development for distributed fuel management in avionics (2013) Journal of Aerospace Computing, Information and Communication, 10 (2), pp. 71-86.

SC122 Viaud, B., Labrèche, P. Citrus: Model-based avionics development with zest! (2013) SAE Technical Papers, 7.

SC129 Wedzinga, G., Wiegmink, K. Using CHARTER tools to develop a safety-critical avionics application in java (2012) ACM International Conference Proceeding Series, pp. 125-134.

SC130 Conrad, M., Englehart, M., Erkkinen, T., Lin, X., Nirakh, A.R., Potter, B., Shankar, J., Szpak, P., Yan, J., Clark, J. Automating code reviews with simulink code inspector (2012) Tagungsband - Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme VIII, MBEES 2012, pp. 31-36.

SC132 Eriksson, A., Lindström, B., Andler, S.F., Offutt, J. Model transformation impact on test artifacts: An empirical study (2012) Proceedings of the Workshop on Model-Driven Engineering, Verification and Validation, MoDeVVa 2012, pp. 5-10.

SC140 Holzer, A., Januzaj, V., Kugele, S., Langer, B., Schallhart, C., Tautschnig, M., Veith, H. Seamless testing for models and code (2011) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6603 LNCS, pp. 278-293.

SC141 Le Sergent, T., Le Guennec, A., Gerard, S., Tanguy, Y., Terrier, F. Using SCADE system for the design and integration of critical systems (2011) SAE Technical Papers.

SC145 Stallbaum, H., Rzepka, M. Toward DO-178B-compliant test models (2010) Proceedings - 2010 Workshop on Model-Driven Engineering, Verification, and Validation, MoDeVVa 2010, art. no. 5772247, pp. 25-30.

DO-178 - Miscellaneous

D002 Paz, A., & El-Boussaidi, G. (2016). On the Exploration of Model-Based Support for DO-178C-Compliant Avionics Software Development and Certification. 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 229-236.

D010 Sarkis, A., & Dias, L.A. (2014). A Set of Rules for Production of Design Models Compliant with Standards DO-178C and DO-331. 2014 11th International Conference on Information Technology: New Generations, 27-32.

D011 Estrada, R.G., Sasaki, G., & Dillaber, E. (2013). Best practices for developing DO-178 compliant software using Model-Based Design.

D012 Pitchai, K.R. (2013). An executable meta-model for safety oriented software and systems development processes within the avionics domain in compliance with RTCA DO 178 B.

D014 Jacklin, S.A. (2012). Certification of Safety-Critical Software Under DO-178C and DO-278A.

D015 Erkkinen, T.J., & Potter, B. (2009). Model-Based Design for DO-178B with Qualified Tools.

EN 50128 - SMS Publications

EN 50128 - Springer

SP002 Gallina B., Gómez-Martínez E., Earle C.B. (2016) Deriving Safety Case Fragments for Assessing MBASafe's Compliance with EN 50128. In: Clarke P., O'Connor R., Rout T., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2016. Communications in Computer and Information Science, vol 609. Springer, Cham

SP003 Svendsen A. et al. (2008) The Future of Train Signaling. In: Czarnecki K., Ober I., Bruel JM., Uhl A., Völter M. (eds) Model Driven Engineering Languages and Systems. MODELS 2008. Lecture Notes in Computer Science, vol 5301. Springer, Berlin, Heidelberg

SP004 Huhn M., Hungar H. (2010) 8 UML for Software Safety and Certification. In: Giese H., Karsai G., Lee E., Rumpe B., Schätz B. (eds) Model-Based Engineering of Embedded Real-Time Systems. MBEERTS 2007. Lecture Notes in Computer Science, vol 6100. Springer, Berlin, Heidelberg

SP005 de la Vara J.L., Panesar-Walawege R.K. (2013) SafetyMet: A Metamodel for Safety Standards. In: Moreira A., Schätz B., Gray J., Vallecillo A., Clarke P. (eds) Model-Driven Engineering Languages and Systems. MODELS 2013. Lecture Notes in Computer Science, vol 8107. Springer, Berlin, Heidelberg

SP007 Luo Y., van den Brand M., Engelen L., Klabbers M. (2014) From Conceptual Models to Safety Assurance. In: Yu E., Dobbie G., Jarke M., Purao S. (eds) Conceptual Modeling. ER 2014. Lecture Notes in Computer Science, vol 8824. Springer, Cham

SP010 Idani A., Ledru Y., Ait Wakrime A., Ben Ayed R., Bon P. (2019) Towards a Tool-Based Domain Specific Approach for Railway Systems Modeling and Validation. In: Collart-Dutilleul S., Lecomte T., Romanovsky A. (eds) Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification. RSSRail 2019. Lecture Notes in Computer Science, vol 11495. Springer, Cham

SP012 Barberio G. et al. (2014) An Interoperable Testing Environment for ERTMS/ETCS Control Systems. In: Bondavalli A., Ceccarelli A., Ortmeier F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2014. Lecture Notes in Computer Science, vol 8696. Springer, Cham

SP016 Hamid B., Geisel J., Ziani A., Gonzalez D. (2012) Safety Lifecycle Development Process Modeling for Embedded Systems - Example of Railway Domain. In: Avgeriou P. (eds) Software Engineering for Resilient Systems. SERENE 2012. Lecture Notes in Computer Science, vol 7527. Springer, Berlin, Heidelberg

SP017 Belmonte F., Soubiran E. (2012) A Model Based Approach for Safety Analysis. In: Ortmeier F., Daniel P. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2012. Lecture Notes in Computer Science, vol 7613. Springer, Berlin, Heidelberg

SP021 Castellanos Ardila J.P., Gallina B., Ul Muram F. (2018) Transforming SPEM 2.0-Compatible Process Models into Models Checkable for Compliance. In: Stamelos I., O'Connor R., Rout T., Dorling A. (eds) Software Process Improvement and Capability Determination. SPICE 2018. Communications in Computer and Information Science, vol 918. Springer, Cham

SP022 Nair S., de la Vara J.L., Melzi A., Tagliaferri G., de-la-Beaujardiere L., Belmonte F. (2014) Safety Evidence Traceability: Problem Analysis and Model. In: Salinesi C., van de Weerd I. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2014. Lecture Notes in Computer Science, vol 8396. Springer, Cham

SP023 Beichler B., Schulz T., Haubelt C., Golatowski F. (2015) A Parametric Dataflow Model for the Speed and Distance Monitoring in Novel Train Control Systems. In: Mousavi M., Berger C. (eds) Cyber Physical Systems. Design, Modeling, and Evaluation. CyPhy 2015. Lecture Notes in Computer Science, vol 9361. Springer, Cham

SP027 Sango M., Duchien L., Gransart C. (2015) Component-Based Modeling and Observer-Based Verification for Railway Safety-Critical Applications. In: Lanese I., Madelaine E. (eds) Formal Aspects of Component Software. FACS 2014. Lecture Notes in Computer Science, vol 8997. Springer, Cham

EN 50128 - Scopus

SC002 Nardone, R., Marrone, S., Gentile, U., Amato, A., Barberio, G., Benerecetti, M., De Guglielmo, R., Di Martino, B., Mazzocca, N., Peron, A., Pisani, G., Velardi, L., Vittorini, V. An OSLC-based environment for system-level functional testing of ERTMS/ETCS controllers (2020) Journal of Systems and Software, 161, art. no. 110478.

SC006 Wu, D., Schnieder, E. Scenario-based system design with colored Petri nets: an application to train control systems (2018) Software and Systems Modeling, 17 (1), pp. 295-317.

SC007 Basile, D., ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F., Piattino, A., Trentini, D., Ferrari, A. On the industrial uptake of formal methods in the railway domain: A survey with stakeholders (2018) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11023 LNCS, pp. 20-29.

SC008 De La Vara, J.L., Ruiz, A., Espinoza, H. Recent advances towards the industrial application of model-driven engineering for assurance of safety-critical systems (2018) MODELSWARD 2018 - Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development, 2018-January, pp. 632-641.

SC010 Gallina, B., Gómez-Martínez, E., Benac-Earle, C. Promoting MBA in the rail sector by deriving process-related evidence via MDSafeCer (2017) Computer Standards and Interfaces, 54, pp. 119-128.

SC012 Radermacher, A., Hamid, B., Fredj, M., Ingenierie, S., Profizi, J.-L. Process and tool support for design patterns with safety requirements (2015) ACM International Conference Proceeding Series, 10-14-July-2013, art. no. a8.

SC019 Carnevali, L., Ridi, L., Vicario, E. Putting preemptive Time Petri Nets to work in a V-Model SW life cycle (2011) IEEE Transactions on Software Engineering, 37 (6), art. no. 5680913, pp. 826-844.

SC020 Cancila, D., Terrier, F., Belmonte, F., Dubois, H., Espinoza, H., Gérard, S., Cuccuru, A. SOPHIA: A modeling language for model-based safety engineering (2009) CEUR Workshop Proceedings, 507, pp. 11-25.

EN 50128 - Miscellaneous

D002 Bondavalli, A., & Brancati, F. (2017). Certifications of Critical Systems - The CECRIS Experience.

D003 Sun, P. (2015). Model based system engineering for safety of railway critical systems.

Appendix B

ASIL SysML Profile MDG (XML File)

```
<?xml version="1.0" encoding="windows-1252"?>
<MDG.Technology version="1.0"><Documentation id="1" name="ASIL SysML Profile" version
    ="1.1.5" notes="Started 22.02.20;Including Severity, Controllability and ExposurexA
    ;"/><UMLProfiles><UMLProfile profiletype="uml2">
   <Documentation id="4E54BD59-5" name="Profile ASIL" version="" notes="Profile ASIL"/>
   <Content>
    <Stereotypes>
     <Stereotype name="safetyBlock" notes="" cx="124" cy="70" bgcolor="-1" fontcolor
         ="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="SysML1.4::
         block" baseStereotypes="SysML1.4::block">
      <AppliesTo>
       <Apply type="Class">
        <Property name="isActive" value=""/>
       </Apply>
      </AppliesTo>
      <TaggedValues>
       <Tag name="ASIL" type="enumeration" description="" unit="" values="A,A (A),A (B)
           ,A (C),A (D),B,B (B),B (C),B (D),C,C (C),C (D),D,D (D),QM,QM (A),QM (B),QM (
           C),QM (D),Unclassified" default="Unclassified"/>
       <Tag name="decomposed" type="boolean" description="" unit="" values="true,false"
           default=""/>
      </TaggedValues>
     </Stereotype>
     <Stereotype name="safetyRequirement" notes="" cx="146" cy="70" bgcolor="-1"
         fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="
         SysML1.4::requirement" baseStereotypes="SysML1.4::requirement">
      <AppliesTo>
       <Apply type="Requirement"/>
      </AppliesTo>
      <TaggedValues>
       <Tag name="ASIL" type="enumeration" description="" unit="" values="A,A (A),A (B)
           ,A (C),A (D),B,B (B),B (C),B (D),C,C (C),C (D),D,D (D),QM,QM (A),QM (B),QM (
           C),QM (D),Unclassified" default="Unclassified"/>
      </TaggedValues>
     </Stereotype>
     <Stereotype name="functionalSafetyRequirement" notes="" cx="148" cy="78" bgcolor
         ="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
         generalizes="safetyRequirement" baseStereotypes="safetyRequirement">
      <TaggedValues>
       <Tag name="decomposed" type="boolean" description="" unit="" values="true,false"
           default=""/>
      </TaggedValues>
     </Stereotype>
     <Stereotype name="safetyGoal" notes="" cx="156" cy="104" bgcolor="-1" fontcolor
         ="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="
         safetyRequirement" baseStereotypes="safetyRequirement">
      <TaggedValues>
       <Tag name="con" type="enumeration" description="" unit="" values="Unclassified,
           C0,C1,C2,C3" default="Unclassified"/>
       <Tag name="exp" type="enumeration" description="" unit="" values="Unclassified,
           E0,E1,E2,E3,E4" default="Unclassified"/>
       <Tag name="sev" type="enumeration" description="" unit="" values="Unclassified,
           S0,S1,S2,S3" default="Unclassified"/>
```

```
      </TaggedValues>
     </Stereotype>
     <Stereotype name="abstractTechnicalSafetyRequirement" notes="" cx="177" cy="94"
          bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
          isAbstract="true" generalizes="functionalSafetyRequirement" baseStereotypes="
          functionalSafetyRequirement"/>
     <Stereotype name="hardwareSafetyRequirement" notes="" cx="148" cy="94" bgcolor
          ="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
          generalizes="abstractTechnicalSafetyRequirement" baseStereotypes="
          abstractTechnicalSafetyRequirement"/>
     <Stereotype name="softwareSafetyRequirement" notes="" cx="144" cy="94" bgcolor
          ="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
          generalizes="abstractTechnicalSafetyRequirement" baseStereotypes="
          abstractTechnicalSafetyRequirement"/>
     <Stereotype name="assignedTo" notes="" cx="107" cy="70" bgcolor="-1" fontcolor
          ="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
      <AppliesTo>
       <Apply type="Dependency">
        <Property name="direction" value="Source -&amp;gt; Destination"/>
       </Apply>
      </AppliesTo>
     </Stereotype>
     <Stereotype name="decomposedFrom" notes="" cx="104" cy="70" bgcolor="-1" fontcolor
          ="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
      <AppliesTo>
       <Apply type="Dependency">
        <Property name="direction" value="Source -&amp;gt; Destination"/>
       </Apply>
      </AppliesTo>
     </Stereotype>
     <Stereotype name="safetyProperty" notes="" cx="124" cy="70" bgcolor="-1" fontcolor
          ="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
      <AppliesTo>
       <Apply type="Part">
        <Property name="isReference" value="false"/>
       </Apply>
      </AppliesTo>
      <TaggedValues>
       <Tag name="ASIL" type="enumeration" description="" unit="" values="A,A (A),A (B)
            ,A (C),A (D),B,B (B),B (C),B (D),C,C (C),C (D),D,D (D),QM,QM (A),QM (B),QM (
           C),QM (D),Unclassified" default="Unclassified"/>
      </TaggedValues>
     </Stereotype>
     <Stereotype name="softwareSafetyProperty" notes="" cx="135" cy="85" bgcolor="-1"
          fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="
          abstractTechnicalSafetyProperty" baseStereotypes="
          abstractTechnicalSafetyProperty"/>
     <Stereotype name="hardwareSafetyProperty" notes="" cx="136" cy="78" bgcolor="-1"
          fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="
          abstractTechnicalSafetyProperty" baseStereotypes="
          abstractTechnicalSafetyProperty"/>
     <Stereotype name="functionalSafetyProperty" notes="" cx="129" cy="70" bgcolor="-1"
           fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="
          safetyProperty" baseStereotypes="safetyProperty"/>
     <Stereotype name="abstractTechnicalSafetyProperty" notes="" cx="160" cy="70"
          bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
          isAbstract="true" generalizes="functionalSafetyProperty" baseStereotypes="
          functionalSafetyProperty"/>
    </Stereotypes>
    <TaggedValueTypes/>
    <ViewDefinitions/>
    <Metamodel/>
   </Content>
  </UMLProfile>
 </UMLProfiles></MDG.Technology>
```

OCL Constraints

SRQ_1

```
context safetyRequirement
inv: self.ASIL <> 'Unclassified'

context functionalSafetyRequirement
inv: self.ASIL <> 'Unclassified'

context softwareSafetyRequirement
inv: self.ASIL <> 'Unclassified'

context hardwareSafetyRequirement
inv: self.ASIL <> 'Unclassified'
```

SRQ_2

```
context safetyRequirement
inv: self.deriveReqt.oclIsTypeOf(safetyGoal)
```

SRQ_3

```
context functionalSafetyRequirement
inv: self.deriveReqt.oclIsTypeOf(safetyRequirement)
```

SRQ_4

```
context technicalSafetyRequirement
inv: self.allInstances->isEmpty()
```

SRQ_5

```
context softwareSafetyRequirement
inv: self.deriveReqt.oclIsTypeOf(functionalSafetyRequirement)

context hardwareSafetyRequirement
inv: self.deriveReqt.oclIsTypeOf(functionalSafetyRequirement)
```

SRQ_6

```
context functionalSafetyRequirement
inv: self.decomposedFrom.oclIsTypeOf(functionalSafetyRequirement)

context softwareSafetyRequirement
inv: self.decomposedFrom.oclIsTypeOf(softwareSafetyRequirement)

context hardwareSafetyRequirement
inv: self.decomposedFrom.oclIsTypeOf(hardwareSafetyRequirement)
```

SRQ_7

```
context functionalSafetyRequirement
inv: self.decomposedFrom->size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or 'A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'

context hardwareSafetyRequirement
inv: self.decomposedFrom->size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or 'A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'

context softwareSafetyRequirement
inv: self.decomposedFrom->size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or 'A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'
```

SRQ_8

```
context functionalSafetyRequirement
inv 1: self.ASIL = 'A' and self.decomposed = TRUE implies self.decomposedTo
 -> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'A (A)' and d2 = 'QM (A)')
or (d1 = 'QM (A)' and d2 = 'A (A)'))

inv 2: self.ASIL = 'B' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'B (B)' and d2 = 'QM (B)')
or (d1 = 'QM (B)' and d2 = 'B (B)') or (d1 = 'A (B)' and d2 = 'A (B)'))

inv 3: self.ASIL = 'C' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'C (C)' and d2 = 'QM (C)')
or (d1 = 'QM (C)' and d2 = 'C (C)') or (d1 = 'B (C)' and d2 = 'A (C)')
or (d1 = 'A (C)' and d2 = 'B (C)'))

inv 4: self.ASIL = 'D' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'D (D)' and d2 = 'QM (D)')
or (d1 = 'QM (D)' and d2 = 'D (D)') or (d1 = 'C (D)' and d2 = 'A (D)')
or (d1 = 'A (D)' and d2 = 'C (D)') or (d1 = 'B (D)' and d2 = 'B (D)'))

context hardwareSafetyRequirement
inv 1: self.ASIL = 'A' and self.decomposed = TRUE implies self.decomposedTo
 -> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'A (A)' and d2 = 'QM (A)')
or (d1 = 'QM (A)' and d2 = 'A (A)'))

inv 2: self.ASIL = 'B' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'B (B)' and d2 = 'QM (B)')
or (d1 = 'QM (B)' and d2 = 'B (B)') or (d1 = 'A (B)' and d2 = 'A (B)'))

inv 3: self.ASIL = 'C' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'C (C)' and d2 = 'QM (C)')
or (d1 = 'QM (C)' and d2 = 'C (C)') or (d1 = 'B (C)' and d2 = 'A (C)')
or (d1 = 'A (C)' and d2 = 'B (C)'))

inv 4: self.ASIL = 'D' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'D (D)' and d2 = 'QM (D)')
or (d1 = 'QM (D)' and d2 = 'D (D)') or (d1 = 'C (D)' and d2 = 'A (D)')
or (d1 = 'A (D)' and d2 = 'C (D)') or (d1 = 'B (D)' and d2 = 'B (D)'))

context softwareSafetyRequirement
inv 1: self.ASIL = 'A' and self.decomposed = TRUE implies self.decomposedTo
 -> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'A (A)' and d2 = 'QM (A)')
or (d1 = 'QM (A)' and d2 = 'A (A)'))

inv 2: self.ASIL = 'B' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'B (B)' and d2 = 'QM (B)')
or (d1 = 'QM (B)' and d2 = 'B (B)') or (d1 = 'A (B)' and d2 = 'A (B)'))

inv 3: self.ASIL = 'C' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'C (C)' and d2 = 'QM (C)')
or (d1 = 'QM (C)' and d2 = 'C (C)') or (d1 = 'B (C)' and d2 = 'A (C)')
or (d1 = 'A (C)' and d2 = 'B (C)'))

inv 4: self.ASIL = 'D' and self.decomposed = TRUE implies self.decomposedTo
-> forAll (d1, d2 : functionalSafetyRequirement | (d1 = 'D (D)' and d2 = 'QM (D)')
or (d1 = 'QM (D)' and d2 = 'D (D)') or (d1 = 'C (D)' and d2 = 'A (D)')
or (d1 = 'A (D)' and d2 = 'C (D)') or (d1 = 'B (D)' and d2 = 'B (D)'))
```

SRQ_9

```
context functionalSafetyRequirement
inv: self.decomposedTo-> size() => 2 implies self.decomposed = TRUE

context hardwareSafetyRequirement
inv: self.decomposedTo-> size() => 2 implies self.decomposed = TRUE

context softwareSafetyRequirement
inv: self.decomposedTo-> size() => 2 implies self.decomposed = TRUE
```

SRQ_10

```
context functionalSafetyRequirement
inv: self.decomposedTo-> size() >= 2 implies self.decomposed
-> forAll(oclIsTypeOf(functionalSafetyRequirement))

context softwareSafetyRequirement
inv: self.decomposedTo-> size() >= 2 implies self.decomposed
-> forAll(oclIsTypeOf(softwareSafetyRequirement))

context hardwareSafetyRequirement
inv: self.decomposedTo-> size() >= 2 implies self.decomposed
-> forAll(oclIsTypeOf(hardwareSafetyRequirement))
```

SRQ_11

```
context safetyRequirement
inv 1: self.ASIL = 'QM'  implies self.deriveReqt -> forAll(ASIL = 'QM')
inv 2: self.ASIL = 'A'  implies self.deriveReqt -> forAll(ASIL = 'A')
inv 3: self.ASIL = 'B'  implies self.deriveReqt -> forAll(ASIL = 'B')
inv 4: self.ASIL = 'C'  implies self.deriveReqt -> forAll(ASIL = 'C')
inv 5: self.ASIL = 'D'  implies self.deriveReqt -> forAll(ASIL = 'D')

context functionalSafetyRequirement
inv 1: self.ASIL = 'QM'  implies self.deriveReqt -> forAll(ASIL = 'QM')
inv 2: self.ASIL = 'A'  implies self.deriveReqt -> forAll(ASIL = 'A')
inv 3: self.ASIL = 'B'  implies self.deriveReqt -> forAll(ASIL = 'B')
inv 4: self.ASIL = 'C'  implies self.deriveReqt -> forAll(ASIL = 'C')
inv 5: self.ASIL = 'D'  implies self.deriveReqt -> forAll(ASIL = 'D')

context softwareSafetyRequirement
inv 1: self.ASIL = 'QM'  implies self.deriveReqt -> forAll(ASIL = 'QM')
inv 2: self.ASIL = 'A'  implies self.deriveReqt -> forAll(ASIL = 'A')
inv 3: self.ASIL = 'B'  implies self.deriveReqt -> forAll(ASIL = 'B')
inv 4: self.ASIL = 'C'  implies self.deriveReqt -> forAll(ASIL = 'C')
inv 5: self.ASIL = 'D'  implies self.deriveReqt -> forAll(ASIL = 'D')

context hardwareSafetyRequirement
inv 1: self.ASIL = 'QM'  implies self.deriveReqt -> forAll(ASIL = 'QM')
inv 2: self.ASIL = 'A'  implies self.deriveReqt -> forAll(ASIL = 'A')
inv 3: self.ASIL = 'B'  implies self.deriveReqt -> forAll(ASIL = 'B')
inv 4: self.ASIL = 'C'  implies self.deriveReqt -> forAll(ASIL = 'C')
inv 5: self.ASIL = 'D'  implies self.deriveReqt -> forAll(ASIL = 'D')
```

SRQ_12

```
context safetyRequirement
inv 1: self.ASIL = 'A' implies self.satisfy -> forAll (ASIL = 'A'
or ASIL = 'A (B)' or ASIL = 'A (C)' or ASIL = 'A (D)' or ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or ASIL = 'C'
or ASIL = 'C (C)' ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 2 : self.ASIL = 'B' implies self.satisfy -> forAll(ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or  ASIL = 'C'
or ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 3 : self.ASIL = 'C' implies self.satisfy -> forAll(ASIL = 'C'
ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 4 : self.ASIL = 'D' implies self.satisfy
-> forAll (ASIL = 'D' or ASIL = 'D (D)')

context functionalSafetyRequirement
inv 1: self.ASIL = 'A' implies self.satisfy -> forAll (ASIL = 'A'
or ASIL = 'A (B)' or ASIL = 'A (C)' or ASIL = 'A (D)' or ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or ASIL = 'C'
or ASIL = 'C (C)' ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 2 : self.ASIL = 'B' implies self.satisfy -> forAll(ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or  ASIL = 'C'
or ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 3 : self.ASIL = 'C' implies self.satisfy -> forAll(ASIL = 'C'
ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 4 : self.ASIL = 'D' implies self.satisfy
-> forAll (ASIL = 'D' or ASIL = 'D (D)')

context softwareSafetyRequirement
```

```
inv 1: self.ASIL = 'A' implies self.satisfy -> forAll (ASIL = 'A'
or ASIL = 'A (B)' or ASIL = 'A (C)' or ASIL = 'A (D)' or ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or ASIL = 'C'
or ASIL = 'C (C)' ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 2 : self.ASIL = 'B' implies self.satisfy -> forAll(ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or  ASIL = 'C'
or ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 3 : self.ASIL = 'C' implies self.satisfy -> forAll(ASIL = 'C'
ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 4 : self.ASIL = 'D' implies self.satisfy
-> forAll (ASIL = 'D' or ASIL = 'D (D)')

context hardwareSafetyRequirement
inv 1: self.ASIL = 'A' implies self.satisfy -> forAll (ASIL = 'A'
or ASIL = 'A (B)' or ASIL = 'A (C)' or ASIL = 'A (D)' or ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or ASIL = 'C'
or ASIL = 'C (C)' ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 2 : self.ASIL = 'B' implies self.satisfy -> forAll(ASIL = 'B'
or ASIL = 'B (B)' or ASIL = 'B (C)' or ASIL = 'B (D)' or  ASIL = 'C'
or ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 3 : self.ASIL = 'C' implies self.satisfy -> forAll(ASIL = 'C'
ASIL = 'C (C)' or ASIL = 'C (D)' or ASIL = 'D' or ASIL = 'D (D)')
inv 4 : self.ASIL = 'D' implies self.satisfy
-> forAll (ASIL = 'D' or ASIL = 'D (D)')
```

SG_1

```
context safetyGoal
inv: self.assignedTo -> forAll(oclIsKindOf(safetyBlock) or oclIsKindOf(safetyProperty))
```

SG_2

```
context safetyGoal
inv 1: self.ASIL = 'QM'  implies self.assignedTo -> forAll(ASIL = 'QM' )
inv 1: self.ASIL = 'A'  implies self.assignedTo.ASIL -> forAll(ASIL = 'A' )
inv 2: self.ASIL = 'B'  implies self.assignedTo.ASIL -> forAll(ASIL = 'B' )
inv 3: self.ASIL = 'C'  implies self.assignedTo.ASIL -> forAll(ASIL = 'C' )
inv 4: self.ASIL = 'D'  implies self.assignedTo.ASIL -> forAll(ASIL = 'D' )
```

SG_3

```
context safetyGoal
inv: self.ASIL <> 'Unclassified'
```

SG_4

```
context safetyGoal
inv: self.con <> 'Unclassified'
```

SG_5

```
context safetyGoal
inv: self.exp <> 'Unclassified'
```

SG_6

```
context safetyGoal
inv: self.sev <> 'Unclassified'
```

SP_1

```
context safetyProperty
inv: self.ASIL <> 'Unclassified'

context functionalSafetyProperty
```

```
inv: self.ASIL <> 'Unclassified'

context softwareSafetyProperty
inv: self.ASIL <> 'Unclassified'

context hardwareSafetyProperty
inv: self.ASIL <> 'Unclassified'
```

SP_2

```
context functionalSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or ' A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'

context softwareSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or ' A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'

context hardwareSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.ASIL = 'A (A)' or 'A (B)' or 'A (C)'
or ' A (D)' or 'B (B)' or 'B (C)' or 'B (D)' or 'C (C)' or 'C (D)' or 'D (D)'
or 'QM (A)' or 'QM (B)' or 'QM (C)' or 'QM (D)'
```

SP_3

```
context functionalSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.satisfy -> forAll(decomposed = TRUE)
implies self.satisfy -> forAll(oclIsTypeOf(functionalSafetyRequirement))

context softwareSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.satisfy -> forAll(decomposed = TRUE)
implies self.satisfy -> forAll(oclIsTypeOf(softwareSafetyRequirement))

context hardwareSafetyProperty
inv: self.decomposedFrom-> size() = 1 implies self.satisfy -> forAll(decomposed = TRUE)
implies self.satisfy -> forAll(oclIsTypeOf(hardwareSafetyRequirement))
```

SP_4

```
context softwareSafetyProperty
inv: self.satisfy -> forAll(oclIsTypeOf(softwareSafetyRequirement))
```

SP_5

```
context hardwareSafetyProperty
inv: self.satisfy -> forAll(oclIsTypeOf(hardwareSafetyRequirement))
```

SP_6

```
context functionalSafetyProperty
inv: self.satisfy -> forAll(oclIsTypeOf(functionalSafetyRequirement))
```

SP_7

```
context safetyProperty
inv: self.safetyProperty -> notEmpty() implies
self.safetyProperty -> forAll(ASIL = self.ASIL)
context functionalSafetyProperty
inv: self.functionalSafetyProperty -> notEmpty() implies
self.functionalSafetyProperty -> forAll(ASIL = self.ASIL)
context softwareSafetyProperty
inv: self.softwareSafetyProperty -> notEmpty() implies
self.softwareSafetyProperty -> forAll(ASIL = self.ASIL)
context hardwareSafetyProperty
inv: self.hardwareSafetyProperty -> notEmpty() implies self.ASIL implies
self.hardwareSafetyProperty -> forAll(ASIL = self.ASIL)
```

SP_8

```
context functionalSafetyProperty
inv: self.decomposed -> size() => 2 implies self.decomposed
->forAll(oclIsTypeOf(functionalSafetyProperty))
context softwareSafetyProperty
inv: self.decomposed -> size() => 2 implies self.decomposed
-> forAll(oclIsTypeOf(softwareSafetyProperty))
context hardwareSafetyProperty
inv: self.decomposed -> size() => 2 implies self.decomposed
-> forAll(oclIsTypeOf(hardwareSafetyProperty))
```