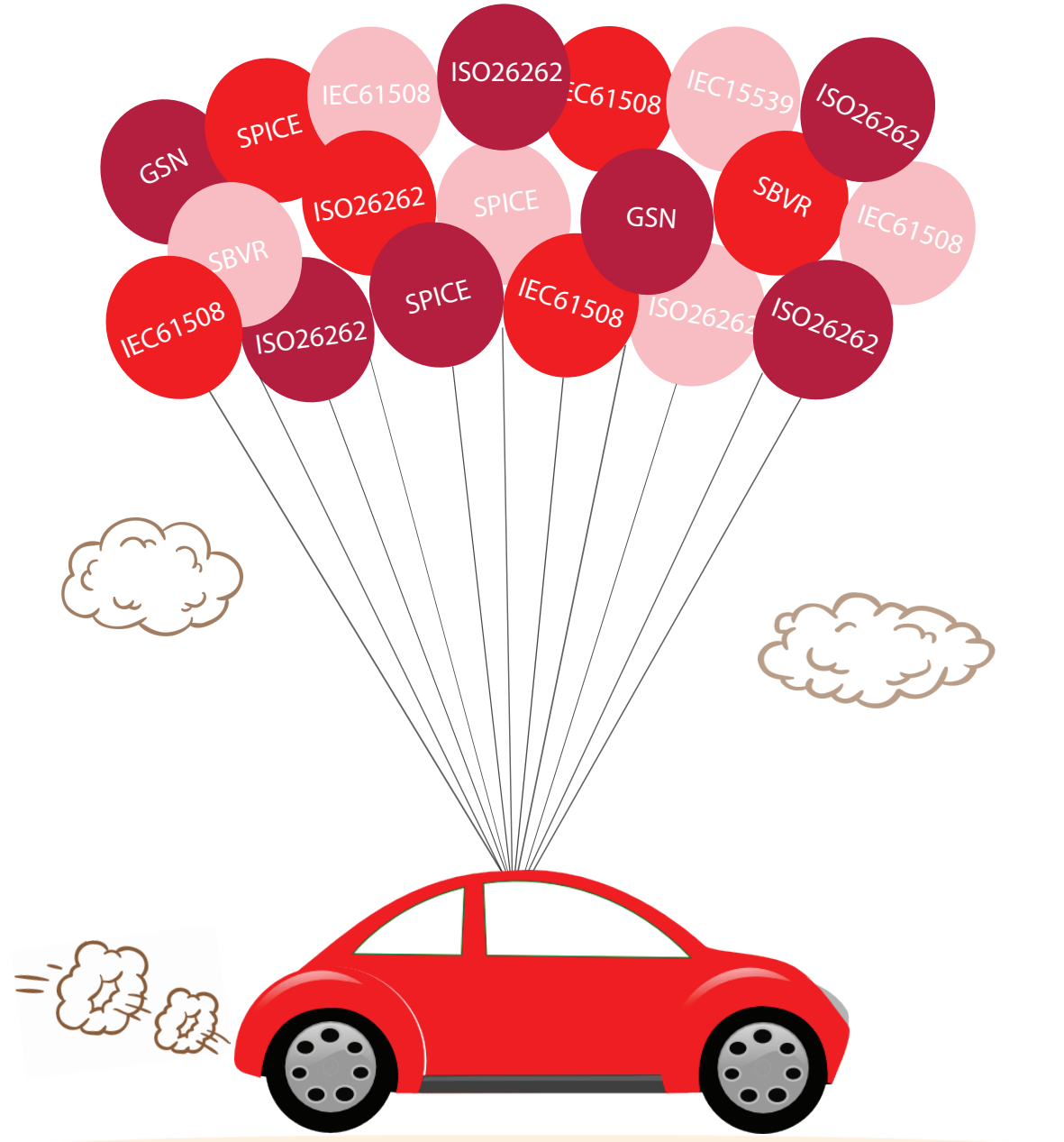
From conceptual models to safety assurance : applying model-based techniques to support safety assurance



From Conceptual Models to Safety Assurance

Applying Model-Based Techniques to Support Safety Assurance

Yaping Luo

From Conceptual Models to Safety Assurance

Applying Model-Based Techniques to Support Safety Assurance

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus, prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op dinsdag 26 april 2016 om 16.00 uur

door

Yaping Luo

geboren te Henan, China

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt: :

voorzitter:	prof.dr.ir. B. Koren
promotor:	prof.dr. M.G.J. van den Brand
copromotoren:	dr. A.Serebrenik
	dr.ir. L.J.P. Engelen (ISAAC Software)
leden:	prof.dr. J.J. Lukkien
	prof.dr. T. Kelly (University of York)
	prof.dr. C. Liu (Beihang University)
adviseur:	dr. J.F.C.M. de Jongh (TNO Delft)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

From Conceptual Models to Safety Assurance

Applying Model-Based Techniques to Support Safety Assurance

Yaping Luo

Promotor:	prof.dr. M.G.J. van den Brand (Eindhoven University of Technology)
Copromotoren:	dr. A.Serebrenik (Eindhoven University of Technology)
	dr.ir. L.J.P. Engelen (ISAAC)

Additional members of the reading committee:

dr. J.F.C.M. de Jongh	(TNO Delft)
prof.dr. J.J. Lukkien	(Eindhoven University of Technology)
prof.dr. T. Kelly	(University of York, UK)
prof.dr. C. Liu	(Beihang University, China)

The work in this thesis has been carried out under the auspices of the research school IPA (Institute for Programming research and Algorithmics).
IPA dissertation series 2016-06.

The work in this thesis has been carried out as part of the Open Platform for EvolutioNary Certification Of Safety-critical Systems (OPENCOSS) project. The OPENCOSS project has been funded by the FP7 programme under grant agreement nº 289011.

A catalogue record is available from the Eindhoven University of Technology Library
ISBN: 978-90-386-4052-5

Cover design: the graphics of cloud and car are originally downloaded from Internet.
For the cloud graphic: `http://www.vecteezy.com/`
For the car graphic: `http://worldartsme.com/`

Printed by Ipskamp Printing

Acknowledgements

Time flies. It has been four years since I became a PhD candidate at TU/e. I still remember when I arrived here, I had a lot of questions in my head about the upcoming four years. Now it turns out I have really enjoyed this journey and it has brought me fruitful experience. For all of these, I am forever grateful to my supervisor Mark van den Brand. He accepted me as one of his students and gave me his expert advice and encouragement throughout my PhD study. During the difficult period of my project, he has even played two roles (supervisor and daily-supervisor) at the same time. Without his support and patience, finishing the PhD journey would only remain as my dream.

At the beginning of my study, Martijn Klabbers and Luc Engelen provided me with guidelines in the OPENCOSS project. With their help, I started to learn how to write project deliverables and scientific papers, discuss with project partners, and give presentations. I am very grateful for their support during these years.

I would also like to express my deep gratitude to Alexander Serebrenik. Before writing this thesis, Alexander agreed to be my co-promoter as well. From that moment on, he helped me review my papers and thesis, and provided me his valuable feedbacks. His support and experience gave me a lot of confidence on finishing this thesis.

In the first three years, I was fully invloved in the OPENCOSS project, which has 17 project partners. I would like to thank all project members for all fruitful discussions and collaborations we have had in the project.

Throughout my project, I have collaborated and coauthor papers with a number of people, which results in a big contribution to this thesis. Besides, Mark, Martjin and Luc, I would like to thank John Favaro (Intecs), Giovanni Sartori (Intecs), S Manoj Kannan, Yanja Dajsuren, Ion Barosan, Alexander Kiburse, Huan Lin (Beihang University), Ji Wu (Beihang University), Chunchun Yuan (Beihang University), Arash Khabbaz Saberi, Filip Pawel Cichosz (TNO), Sven Jansen (TNO), Jaap Stelma, Zhuoao Li. I also want to thank Johan van Uden for his help on the power window case study.

Besides doing research, I have been involved in teaching and supervising master students and PDeng students as well. For this, I would like to thank Mark again for giving me these opportunities. Moreover I would also like to thank Ruurd Kuiper for inviting me to be an instructor in his course.

Furthermore, I would like to extend my sincere gratitude to the members of my reading committee for reviewing this dissertation and providing their valuable feedback:

Jan de Jongh from TNO Delft, Tim Kelly from University of York, Chao Liu from Beihang University, and Johan Lukkien from Eindhoven University of Technology.

In the first one and half year, I was located at LaQuSo group. I would like to use this opportunity to thank all the former LaQuSo members. I would like to thank Harold Weffers for welcoming me on my first day here and making me feel like home. Besides, I would like to thank Joost Gabriels, Reinier Post, Serguei Roubtsov, Ion Barosan, Willeke Quaedflieg for all the valuable memories in LaQuSo.

After LaQuSo got discontinued, I moved to SET group and shared an office with Önder Babur. Later on, we relocated and had more officemates: Frank Peter, Rob Faessen, Loek Cleophas, Jia Zhang, Zhuoao Li, Jaewon Oh, Alexander Aroyo, and Miguel Botto Tobar. I enjoyed the time we spent together, many thanks to all of them for this. Besides I also want to thank other past and current colleagues, especially (in no particular order) Ana-Maria Sutii, Dan Zhang, Neda Noroozi, Yanja Dajsuren, Ulyana Tikhonova, Bogdan Vasilescu, Yuexu Chen, Maarten Manders, Sander de Putter, Josh Mengerink, Ali Afroozeh, Margje Mommers-Lenders, Tineke van den Bosch, Eric Scheffers, Dragan Bosnacki, Anton Wijs, Tom Verhoeff, Kees Huizing, Fei Yang, Sarmen Keshishzadeh, Maggy de Wert, Tim Willemse, Sjoerd Cranen, Mohammad Gharehyazie, Jan Friso Groote.

I would also like to thank my dance group members, all COSMOS members and other friends. They made my spare time full of joy.

Finally, I would like to thank my family and my boyfriend Valcho Dimitrov for their continuous support and love.

Yaping Luo
Eindhoven, February 2016

Table of Contents

List of Acronyms

ASIL	Automotive Safety Integrity Level
BPMN	Business Process Model and Notation
CCL	Common Certification Language
EMF	Eclipse Modeling Framework
EPF	Eclipse Process Framework
FMEA	Failure Mode and Effects Analysis
FSC	Functional Safety Concept
FSR	Functional Safety Requirements
FTA	Fault Tree Analysis
GMF	Graphical Modeling Framework
GMM	Generic MetaModel
GQM	Goal Question Metric
GSN	Goal Structuring Notation
HARA	Hazard Analysis and Risk Assessment
MOF	Meta-Object Facility
PSM	Practical Software and Systems Measurement
RTOS	Aircraft Real-Time Operating System
SBVR	Semantics of Business Vocabulary and Business Rules
SCCM	Safety Case Conceptual Model
SOCM	Safety Objective Conceptual Model
TSR	Technical Safety Requirements
UML	Unified Modeling Language
V&V	Verification and Validation

Chapter 1

Introduction

In safety-critical domains, such as the automotive, railway, and avionics domains, failure or malfunction of a safety-critical system may result in death or serious injuries to people, as well as severe damage to equipment. Manufacturers in those domains are expected to deliver continuously-safe products. From the end of 2009 to start of 2010, Toyota recalled millions of vehicles that were potentially prone to uncontrolled acceleration. Toyota announced that the company could face losses around US$2 billion from lost sales worldwide [10]. In July 2011, two high-speed trains collided on a viaduct in the suburbs of Wenzhou, Zhejiang province, China. In total 40 people were killed, at least 192 were injured, 12 of which suffered severe injuries. This accident was caused by a faulty signal system which failed to warn the second train of the stationary first train on the same track [13].

With the increasing complexity of software-intensive safety-critical embedded systems, more and more effort is needed to ensure their safety. Over the last two decades, a number of international functional safety standards have been developed to provide development guidelines and keep the risk at an acceptable level [36], such as IEC 61508 (multiple domains) [11], ISO 26262 (automotive domain) [12], DO 178C (avionic domain) [14], CENELEC railway standards (railway domain) [45] [47] [46], etc. Those standards are typically large documents containing a huge number of requirements for system development. The safety standards describe generalized approaches to identifying hazards and risks, design life-cycles, and analysis and design techniques. Therefore, when applying such standards to a specific application, a significant degree of interpretation may be necessary. Besides, adherence to such standards is often the basis for safety assurance and certification (see Section 1.2.1). Safety standards for different domains may focus on different aspects. For example, in the automotive domain, the ISO 26262 standard is a product-based standard focusing on functional safety, while SPICE (ISO/IEC 15504) [60] is a process-based standard focusing on software process assessment. In the avionic domain, DO 178C deals with the safety of software used in certain airborne systems, while ARP 4754 [134] deals with the development processes of aircraft systems. In the railway domain, the European EN 5012x family of railway standards including

EN 50126 [45], EN 50128 [47] and EN 50129 [46] have been developed by CENELEC (European Committee for Electro-technical Standardization). These standards can apply to heavy rail systems, light rail systems, and urban mass transportation including people mover systems. Furthermore, system suppliers may use more than one standard for their products. Company-specific regulations can be based on a number of safety standards.

In some international safety standards, explicit safety cases are highly recommended for safety-critical systems. For example, the ISO 26262 [12] in the automotive domain and the EN 50129 in the railway domain stimulate the use of safety cases to demonstrate the product safety [126]. Besides, the MOD Def Stan 00-55 [106] for safety-critical software in defense equipment requires producing safety cases with explicit safety requirements. A safety case is defined as: "A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment" [42]. In other words, a safety case is used for assuring the desired safety and showing the confidence in the claimed safety assurance. Therefore, it must be trustworthy and understandable. If the context of a safety case is not clear, it may affect its correctness and confidence. Typically safety cases are manually constructed and assessed.

Due to the extensive manual work required, validating compliance of systems with safety standards and constructing safety cases are expensive and time-consuming activities; furthermore, as products evolve, re-assessment and modification may become necessary. Consequently, safety assurance or certification becomes one of the most costly and time-consuming tasks. Moreover, different transport sectors have developed their own specific sets of safety standards, which creates a big challenge to reuse pre-certified components and share expertise between different transport sectors. Recent years, a number of European projects worked on addressing these challenges, such as SafeCer (Safety Certification of Software-Intensive Systems with Reusable Components) [30] and ASCOS (Aviation Safety and Certification of new Operations and Systems) [29].

1.1 Project Objectives

The work described in this thesis was initiated by the OPENCOSS project [61]. This project is an FP7 large-scale integrated project, which started in October 2011 an ended in March 2015 consisting of a consortium of seventeen companies from nine countries. It aims 1) at developing a common certification framework which spans different vertical markets in the transport sector, such as railway, avionics and automotive industries, and facilitates the reuse of assurance assets within, across, and between domains, and 2) establishing an open-source platform as safety certification infrastructure. The ultimate goal of the project was to bring substantial reductions in recurring costs of safety (re-)certification, and at the same time to increase product safety through the introduction of more systematic certification practices. Both are expected to boost innovations and system upgrades considerably. Using a common conceptual framework for different certification standards allows patterns of certification assessment to be shared, and supports cost-effective re-certification between different standards.

In particular, the core challenge of the OPENCOSS project was to define a Common Certification Language (CCL) for specifying certification assets, which works as a means to get mutual recognition agreement between two different conceptual approaches: standard-based approach and argument-based approach [19]. In the OPENCOSS deliverable D4.2 [20] and D4.3 [22], the description and the goal of both approaches are given:

- ***The standard-based approach****: indicating a level of rigor in the processes and*

workflows used to build the final system and specifying the intermediate artifacts to be produced (requirements, specifications, test plans, etc.), the kinds of reviews, analyses that should be performed, and the documentation that should record all of these. The standard-based approach enables us to perform informed gap analysis on the standards, and mitigate against the danger of inappropriately reusing an artefact which does not match the requirements of the reuse context appropriately.

- ***The argument-based approach**: promoting safety certification as a judgment based on a body of material that should consist of claims, evidence and argument. The argument makes the case, based on the evidence, that the claims are satisfied. In the argument-based approach, understanding of the objectives enables us to state clearly what can be claimed for a given artifact and accounted for in the reuse domain.*

During the development of the CCL, two kinds of conceptual models are proposed: a compositional conceptual model and a generic conceptual model. The idea of a compositional conceptual model is to cover all the concepts from three safety-critical domains (automotive, railway, and avionics). However, same concept might have different interpretations and usages for different domains or companies. This makes the compositional conceptual model too big and complex to be useful. Therefore, a generic conceptual model becomes more promising, because it contains the common concepts between those domains. A concrete example of this is a Generic MetaModel (GMM) of safety standards, the benefits of which are: patterns of certification assessment can be shared, and cost-effective re-certification between different standards is supported [145] [19].[1]

As mentioned before, there are two main project goals and results: standard-based and argument-based approach. This thesis contributes to both. For the standard-based approach, the target is to standardize the abstract notations from different domains of safety-critical systems. As the result, a metamodel of the common certification language is defined by domain experts from different project partners [19]. One goal of the CCL is to build domain-specific libraries of certification metamodels, which will act as a knowledge base providing information about safety-related standards. This thesis contributes to this goal by presenting a rule-based solution for modeling and analyzing safety standards (see Chapter 2). Besides, we also carried out research on the evolution and refinement of the CCL (see Chapter 3 and Chapter 4). For the argument-based approach, the target is to provide a structured way for argumentative reasoning about safety requirements and constraints across multiple schemes. This thesis contributes to this purpose by using a controlled language to construct safety cases to reduce the ambiguities that are usually present in them (see Chapter 5). Moreover, the evidence collection for safety cases has also been studied (see Chapter 6). Finally, another contribution of this thesis can be found on using the ISO 26262 to facilitate functional safety management of existing systems (see Chapter 7). This thesis also contributes to the safety certification infrastructure, which is built based on the CCL, by providing metric support for safety assurance process (see Chapter 8).

[1] The GMM used in this thesis is mainly based on the published version [19] with some changes according to our GMM implementation.

1.2 Background

In this section, we introduce a number of basic concepts used in this thesis. We briefly describe safety assessment and safety certification, metamodeling, as well as conceptual modeling.

1.2.1 Safety Assessment and Safety Certification

Safety-critical systems are often required to undergo a stringent safety assessment procedure to show that they meet the required safety objectives. The basic objective of safety assessment is to determine whether the submissions of the operators or developers demonstrate that an activity complies with the stipulated safety objectives or requirements. In other words, safety assessment aims at checking that a safety-critical system complies with the safety objectives throughout its lifetime.

Typically in the safety-critical domains, a formal safety assessment process is required to be carried out by an independent organization. This is often referred to as safety assurance or safety certification. The process of safety certification always includes material reviews, testing and facility inspection, which covers the assessment of the product/system, as well as the processes and personnel involved in the development. The goal of safety certification is to check that the final product/system complies with specific standards for safety, quality or performance.

In the automotive domain, the ISO 26262 standard does not require certification by an assessment authority. In this thesis, we use the concept "safety assessment" for the approaches and case studies in the automotive domain.

1.2.2 Metamodeling

In software engineering, models are more and more widely used. A model always conforms to a unique metamodel, which conforms to a metametamodel. Multilevel metamodeling is designed to avoid many typical problems of metamodeling, such as ambiguous classification and the replication of concepts [72]. Currently, there are two well-known architectures for multilevel metamodeling: the metamodeling architecture of OMG [117] and the Eclipse Modeling Framework (EMF) [2]. Both architectures introduce four levels (Figure 1.1) and use a linguistic classification [37] to divide elements over these levels. Linguistic classification divides elements into two groups: groups of instances and groups of types. Each element of the first group is an instance of an element of the second group. The types of all of the instances on a given level reside on the nearest higher level and describe the properties of the instances.

Atkinson and Kühne identify a second, orthogonal dimension of metamodeling [37]. They state that the dimension of linguistic classification is concerned with language

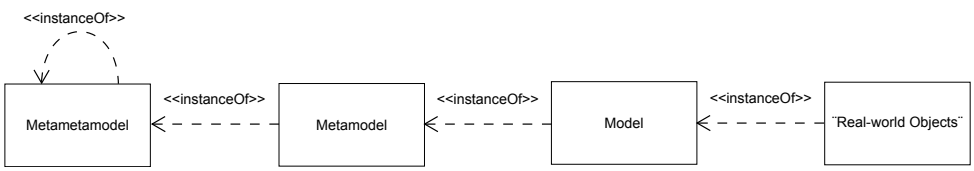


Figure 1.1: Traditional multilevel metamodeling architectures introduce four levels: the metameta level, the meta level, the model level, and the object level.

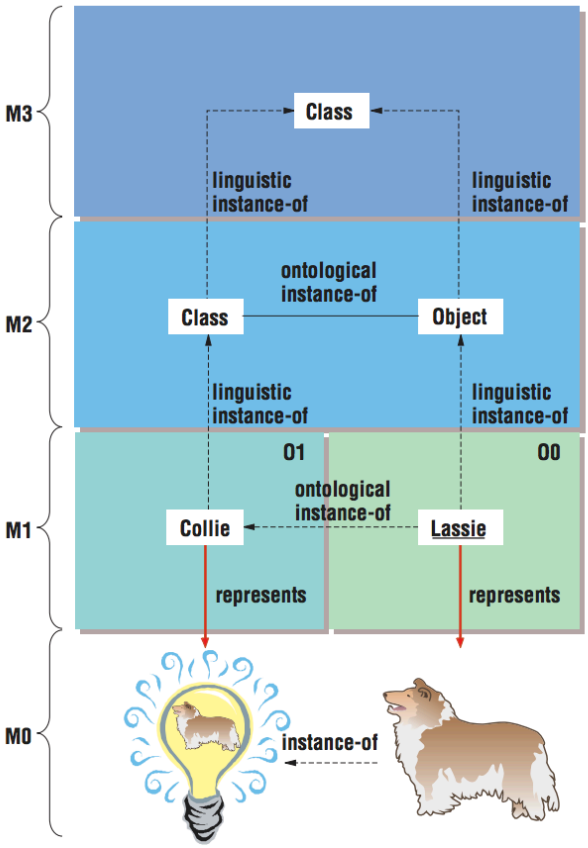


Figure 1.2: "Linguistic metamodeling view" (Figure 2 from [37]). This figure shows the interpretation of the four-layer MOF architecture with UML 2.0. The light bulb represents the mental concept "Collie".

definition and makes use of linguistic instantiation. The other dimension is concerned with domain definition and uses ontological instantiation. Figure 1.2 shows a linguistic metamodeling view. Linguistic instantiation crosses the linguistic metalevels, for example the relationship between Class and Collie. Ontological instantiation, on the other hand, does not cross the linguistic metalevels, but relates elements on the same level instead, for instance the relationship between Collie and Lassie. In other words, ontological instantiation makes it possible to specify that an element on a given level is an instance of an element at the same level.

The approaches described in this thesis strictly focus on linguistic instantiation. Although we appreciate the usefulness of the other dimension of metamodeling, our aim is to investigate the possibilities offered by the more traditional metamodeling approach first. An advantage of doing so is the fact that popular implementations of frameworks for metamodeling as such support linguistic instantiation.

1.2.3 Conceptual Modeling

A conceptual model (also known as domain model) is an abstract representation of things in the real world [141]. Conceptual models enhance models with concepts that are commonly shared within a community or at least between the stakeholders involved in the modeling process [142]. The aim of a conceptual model is to express the meaning of terms and concepts used by experts to discuss problems in their domain, and to find the correct relationships between different concepts. In this way, the meaning of various terms can be clarified and a common understanding of these concepts among experts can be reached. Within information technology, conceptual modeling can be used as the first step for application engineering and support human knowledge integration and global communication [49]. A conceptual model can be expressed in different notations, such as UML [143], OWL [35] and EMF. All of these notations can be used to define the concepts used to describe a domain and the relationships that hold between them. However, the representation of these notations are different. In UML notation, a conceptual model is described as a class diagram. In OWL notation, a conceptual model is represented as an ontology [76] which consists of a set of classes or terms with relations that operate between them. In EMF notation, a conceptual model is created as an Ecore diagram where classes represent concepts and references represent relations between concepts.

Normally, metamodels also consist of concepts and their relations. However, metamodels are usually designed for describing models, and then used for implementation of tools, while conceptual models are independent of design or implementation concerns. Typically a conceptual model is made of a large amount of concepts, while a metamodel is not. Moreover, a metamodel can be derived from or used as a conceptual model. For example, in this thesis, we first work on extracting conceptual models of safety standards. Then the metamodels of safety standards are refined from these conceptual models. Finally, these metamodels are used for tool implementation and model construction.

1.3 Research Questions

We formulated a number of research questions aimed at contributing to the project objectives described in Section 1.1. The central research question is as follows.

> **RQ:** *Can model-driven techniques support the current safety assurance processes?*

This central research question is split into six more specific research questions, RQ_1 through RQ_6. Each of these questions is addressed in the remainder of this thesis.

As compliance with safety standards is a crucial part of safety assurance, we started our research by studying the modeling of safety standards. We noticed that all models of safety standards that were publicly available are built manually by domain experts. The traceability of the source of the models is not maintained. To be able to model safety standards in a systematic way, we posed the following research question.

> **RQ_1:** *How can models for safety standards be created efficiently?*

As the project was progressing, the GMM was built by domain experts from project partners. During the development of the GMM, we noticed that when companies chose to use the GMM, their current way of working had to be changed to conform to the GMM. If they want to build a specific metamodel by introducing new concepts to the GMM, a

refinement process will be needed. Moreover, as new version of safety standards might be proposed in the future, the GMM will need to evolve as well. To support evolution and change of the GMM and metamodels in general, we designed a domain specific language for metamodel refinement. The following research question is related to this.

> **RQ_2:** *How to support GMM evolution and refinement for different safety domains?*

The usage of safety cases is stimulated by several safety standards to demonstrate product safety [12] [46]. Therefore, it should be trustworthy and understandable. Safety cases are expressed in natural languages, which are unavoidably ambiguous. Therefore, we investigated using a controlled language to help safety case construction. The following research question is related to this study.

> **RQ_3:** *Can we reduce the ambiguities in the safety case by using a controlled language?*

While building safety cases, safety evidence needs to be collected to indicate that the safety objectives have been met. After expressing safety cases in controlled language, we investigated using safety claims to facilitate safety evidence collection. This led to the following research question.

> **RQ_4:** *How to collect the required evidence for safety claims efficiently?*

Safety standards in the automotive domain, such as ISO 26262, have become widely used for safety assurance. Automotive companies start to use ISO 26262 as a guideline for their system development. However, ISO 26262 does not deal with the improvement of existing systems, which have not been developed following the standard. Therefore, applying ISO 26262 for functional safety management of an existing system is a challenge. In search of a practical solution to this problem, we formulated the following question.

> **RQ_5:** *How to manage functional safety according to ISO 26262 for an existing safety-critical system?*

Due to the manual work involved, safety assessment processes are costly, time consuming, and hard to estimate. Metrics for safety assessment can, for instance, identify costly activities or phases. Thus, we investigated how to design metrics to evaluate the overall cost and monitor the safety assurance process.
The following question is aimed at metric design according to safety standards.

> **RQ_6:** *How to design metrics for safety assessment process?*

1.4 Outline and Origin of Chapters

The remainder of this thesis is structured as follows. For each chapter that is based on an earlier publication, the origin of the chapter is given. All of the original work has been revised for this thesis to reflect our growing insight.

Chapter 2: Modeling Safety Standards In this chapter, we address research question RQ_1. We illustrate how to extract models from safety standards. The automotive safety standard ISO 26262 is used as a specific example. The validation of our approach is carried out in the context of the OPENCOSS project. This chapter is based on the following publications.

[97] Y. Luo, M.G.J. van den Brand, L.J.P. Engelen, J.M. Favaro, M. Klabbers and G. Sartori. Extracting Models from ISO 26262 for Reusable Safety Assurance. *Proceedings of the 13th International Conference on Software Reuse*, page 192-207. Springer, 2013.

[84] S.M. Kannan, Y. Dajsuren, Y. Luo, and I. Barosan. Analysis of ISO 26262 Compliant Techniques for the Automotive Domain. *Proceedings of International Workshop on Modelling in Automotive Software Engineering*, page 33-42. CEUR, 2015.

Chapter 3: Metamodeling for Safety Assurance In this chapter, we address research question RQ_2 by introducing a metamodel transformation approach to derive domain or project specific metamodels using a generic metamodel as basis. A metamodel refinement language is designed to support the metamodel transformation. We use two automotive case studies as demonstration. This chapter is based on the following publication.

[102] Y. Luo, M.G.J. van den Brand, L.J.P. Engelen, and M. Klabbers. From Conceptual Models to Safety Assurance. *Proceedings of the 33rd International Conference on Conceptual Modeling (ER 2014)*, page 195-208. Springer, 2014.

Chapter 4: Traceability Management during Metamodel Transformation In this chapter, we address research question RQ_2 by discussing how to achieve traceability management during the aforementioned metamodel transformation. This chapter is based on the following publication.

[94] Y. Luo, L.J.P. Engelen, and M.G.J. van den Brand. Metamodel Comparison and Model Comparison for Safety Assurance. *Proceedings of the 3rd International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems*, page 419-430. Springer, 2014.

Chapter 5: Using SBVR for Safety Case Construction In this chapter, we address research question RQ_3 by expressing safety cases in SBVR controlled language. To support this, an SBVR vocabulary editor has been developed and integrated with a safety case editor. This chapter is based on the following publications.

[99] Y. Luo, M.G.J. van den Brand, L.J.P. Engelen, and M.D. Klabbers. A Modeling Approach to Support Safety Assurance in the Automotive Domain. *Proceedings of the Twenty-Third International Conference on Systems Engineering*, page 339-345. Springer, 2015.

[98] Y. Luo, M.G.J. van den Brand, L.J.P. Engelen, and M.D. Klabbers. A Modeling Approach to Support Safety Assurance in the Automotive Domain. *Proceedings of the FISITA World Automotive Congress*, page 1-13. FISITA, 2014.

[100] Y. Luo, M.G.J. van den Brand, and A. Kiburse. Safety Case Development with SBVR-based Controlled Language. *Proceedings of Third International Conference on Model-Driven Engineering and Software Development*, page 1-15. Springer, 2015.

[101] Y. Luo, M.G.J. van den Brand, and Z. Li. A Categorization of GSN-based Safety Cases and Patterns. *Proceedings of Fourth International Conference on Model-Driven Engineering and Software Development*, accepted. 2015.

Chapter 6: Safety Evidence Collection In this chapter, we address research question RQ_4. We describe a systematic approach to facilitate safety evidence collection according to safety claims. Besides, an industrial case study on an avionics Real-Time Operating System is conducted. This chapter is based on the following publication.

[93] H. Lin, Y. Luo, J. Wu, C. Yuan, M.G.J. van den Brand, and L.J.P. Engelen. A Systematic Approach for Safety Evidence Collection in the Safety-Critical Domain. *Proceedings of the 8th Annual IEEE System Conference (SysCon)*, page 194-199. IEEE, 2015.

Chapter 7: Functional Safety Management According to ISO 26262 In this chapter, we address research question RQ_5 by applying ISO 26262 to an existing system which has already been developed using conventional safety methods. To do it, a case study is done in the context of automated driving. This chapter is based on the following publication.

[133] A.K. Saberi, Y. Luo, F.P. Cichosz, M.G.J. van den Brand, and S. Jansen. An Approach for Functional Safety Improvement of an Existing Automotive System. *Proceedings of the 8th Annual IEEE System Conference (SysCon)*, page 277-282. IEEE, 2015.

Chapter 8: Safety Metrics Design for Safety Assurance In this chapter, we address research question RQ_6. We describe a method that makes it possible to design metrics according to the ISO 26262 standard. These metrics can identify costly processes in the safety assurance process. This chapter is based on the following publications.

[95] Y. Luo, J. Stelma, and M.G.J. van den Brand. Functional Safety Measurement in the Automotive Domain: Adaptation of PSM. *Proceedings of the First International Workshop on Automotive Software Architecture*, page 11-17. ACM, 2015.

[96] Y. Luo, and M.G.J. van den Brand. Metrics Design for Safety Assessment. *Information and Software Technology (Accepted)*, 2015.

Chapter 9: Conclusions This final chapter concludes this thesis. It revisits the research questions and gives directions for future research.

1.5 Suggested Method of Reading

Figure 1.3 shows the structure overview of this thesis. Chapter 2 to 4 focus on the safety standard-based approach. Chapter 5 and 6 focus on the safety case-based approach. Chapter 7 and Chapter 8 focus on the overall safety assurance process. They are largely self-contained and can be read independently from each other.

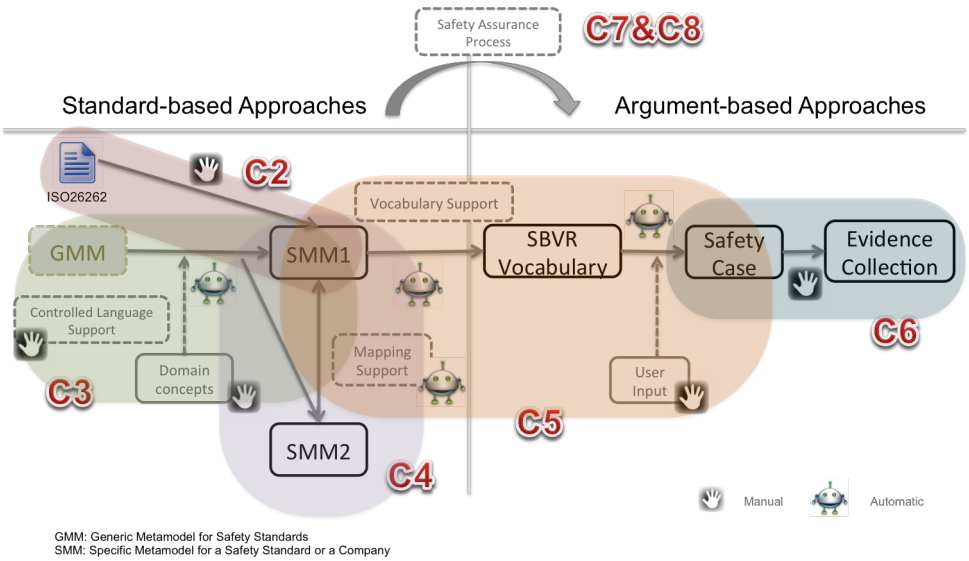


Figure 1.3: Overview of the structure of this thesis.

Chapter 2

Modeling Safety Standards

In this chapter, we propose a model-based approach for assuring compliance with safety standards to facilitate reuse in the assessment, qualification and certification processes, using the automotive safety standard ISO 26262 as a specific example. Three different modeling techniques are described: A structure model is introduced to describe the overall structure of the standard; a rule-based technique is used for extracting the conceptual model from it; and a mapping to the software and systems process engineering metamodel provides a description of its processes. Finally, validation in the context of a concrete use case in the FP7 project OPENCOSS shows that the resulting models of our approach resemble the industrial models, but that they, inevitably, require the fine-tuning of domain experts.

2.1 Introduction

With the increasing complexity of software-intensive safety-critical embedded systems, more and more effort is necessary to ensure their safety. A number of international functional safety standards have been developed to provide development guidelines and keep the risk at an acceptable level [36]. Adherence to such standards is the basis for safety assurance and certification[1]. However, current safety assessment practices are among the most costly and time consuming tasks in development of safety-critical embedded systems [82]; moreover, re-assessment is often needed whenever such systems evolve. Therefore, it is crucial to find an efficient way to reduce the recurring safety assessment costs. This motivates the use of models that capture these safety standards and support the reuse of assurance artifacts. In some domains (such as the automotive domain), there is no authority providing an interpretation of the safety standard, and the modeling process is mainly performed by experts based on manufacturer requirements to ensure sufficient quality. Thus, the whole process of extracting models from the safety standards

[1] In the remainder of this chapter, we mention safety assessment only, because the ISO 26262 standard does not require certification by an assessment authority. Furthermore, for the definitions of the safety-related concepts in this chapter, we refer to the OPENCOSS Deliverable D2.2 [18].

becomes subjective. Furthermore, when a new version of the standard is released, the models need to be updated or modified by persons who may not yet have the competencies. Due to the invisible modeling process, most of the previous work needs to be redone. Hence, finding an objective way for creating the models is crucial.

The motivation for safety assurance reuse goes beyond re-assessment *within* product evolution. It can be even more valuable *across* product development efforts. Sometimes, different systems are developed with similar safety-related characteristics, and the assurance artifacts used for one system can be reused (with appropriate modification) in the assurance activities for another system, resulting in considerable savings. In addition, the assurance artifacts themselves become more reliable with repeated reuse. Just as systematic software reuse is facilitated by formal modeling of the software application domain (the "domain model"), *systematic assurance reuse* can benefit from formal modeling of the *standard* that governs the domain. It is the *model* of a safety standard that facilitates systematic reuse in safety assessment.

The idea of modeling standards is not new: modeling of safety standards is widely used for understanding and communication among engineers and software developers. Besides the aforementioned challenge concerning the subjectiveness of the modeling process, there are other significant challenges to deal with. First, standards are invariably represented in natural language, with the resulting inevitable manual work of interpretation becoming more costly and less reliable. It also increases the difficulty of identifying the reusable information from the safety-related artifacts developed during the safety lifecycle. Second, standards themselves contain inconsistencies. There are a number of synonyms used in the standard, which makes it impossible to generate the models from the standards automatically. Sometimes, standards are even in contradiction with themselves [139]. For example, in ISO 26262, formal methods are merely recommended, while the use of semi-formal methods is always highly recommended. However the standard does mention formal methods and formal notations at several places. Finally, any formal model should support the demonstration of compliance with the safety standard, both for the development process and for the diverse artifacts created during product development. We advocate that standards need to be universally understandable and expressed in a language that is simple, well structured, but strict. For this goal, we believe that in the future it should be possible to transform standards into models automatically, and vice versa.

Work to date has generally involved conceptual modeling of standards for understanding. A conceptual model for the aeronautic standard DO 178B [8] is created to improve communication and collaboration among safety engineers and software engineers [153]. A conceptual model of the generic standard IEC 61508 [11] for electrical and electronic equipment is proposed for the development of compliant embedded software [123]. Also, a study on process modeling has been done in the context of ISO 26262 [90]. All of these studies refer to compliance with the standards from a specific point of view; moreover, the modeling process is subjective, which may lead to inconsistencies of the models after future modifications. Furthermore, the *traceability* of the source of the models is not covered: no one knows where the concepts and relations in the models come from, except the expert who has identified or defined them. To the best of our knowledge, there does not yet exist a methodology to support extracting a conceptual model from the standard in an objective way, a crucial step along the way to systematic assurance reuse.

In view of these observations, we present a solution in this chapter for creating models of safety standards in an objective manner, with the ultimate goal of using the models for demonstrating compliance and enabling reuse of assurance artifacts. As noted earlier,

compliance demonstration requires two models: the conceptual model and the process model. We have developed the so-called "Snowball" approach for extracting the conceptual model from the standard. It is a rule-based approach that reduces the amount of manual work and provides traceability between the model and the standard. For the process model, we use a mapping between the standard and a general process model, providing another way to reduce the subjectivity of the modeling process. ISO 26262, the recent standard for functional safety in the automotive domain, is used as a specific example in this chapter; however, this work also forms the basis for facilitating a more general mapping between standards in different domains.

The rest of the chapter is organized as follows: Section 2.2 reviews the ISO 26262 standard, and describes the proposed modeling approach for enabling safety assurance reuse in the context of OPENCOSS project. In Section 2.3, a rule-based approach is defined for extracting a conceptual model. Section 2.4 outlines a process model of the ISO 26262 *Concept Phase*. Section 2.5 discusses the validation of those models. Section 2.6 introduces the related work of safety assurance reuse. Finally, Section 2.7 summarizes this chapter.

2.2 Background

This section provides the basic information about the ISO 26262 safety standard and the different models derived from ISO 26262.

2.2.1 ISO26262

ISO 26262 is a goal-oriented standard for safety-critical systems within the domain of road vehicles. It is an adaptation of the generic IEC 61508 standard, which focuses on Electrical/Electronic (E/E) systems but provides a general design framework for safety-critical systems [91]. Similar to IEC 61508, ISO 26262 is a risk-based safety standard. It provides a risk-driven safety lifecycle for developing safety-critical systems in the automotive domain. In the standard the risk of hazardous operational situations is qualitatively assessed. This is done to avoid or control systematic failures and to detect or control random hardware failures.

The ISO 26262 consists of ten parts as shown in Figure 2.1. Part 3 to Part 7 correspond to the safety lifecycle, while Part 1, 2, and Part 8 to 10 provides the additional information. The main part of ISO 26262 is structured based upon the V-model, as well as Part 5 and Part 6. Part 3 and Part 7 focus on the vehicle level. The main goal of Part 3 is to identify system hazards and risks through Hazard Analysis and Risk Assessment (HARA), then derive safety goals and Functional Safety Concepts (FSC) from them. Part 4 focuses on the system level. In this part, Technical Safety Requirements (TSR) are derived from FSC. System design can be carried out based on TSR. Part 5 and Part 6 focus on the subsystem/component level. In these two parts more detailed safety requirements are derived from TSR. Those safety requirements are assigned to concrete subsystems or components for implementation.

More and more manufacturers in the automotive domain are developing software-intensive systems in compliance with ISO 26262 [15]. But the safety standard describes generalized approaches for functional safety. When applying the standard, a certain degree of interpretation is needed. Consequently, the efficient deployment of this standard requires changes to the development process in specific projects. Two levels may be

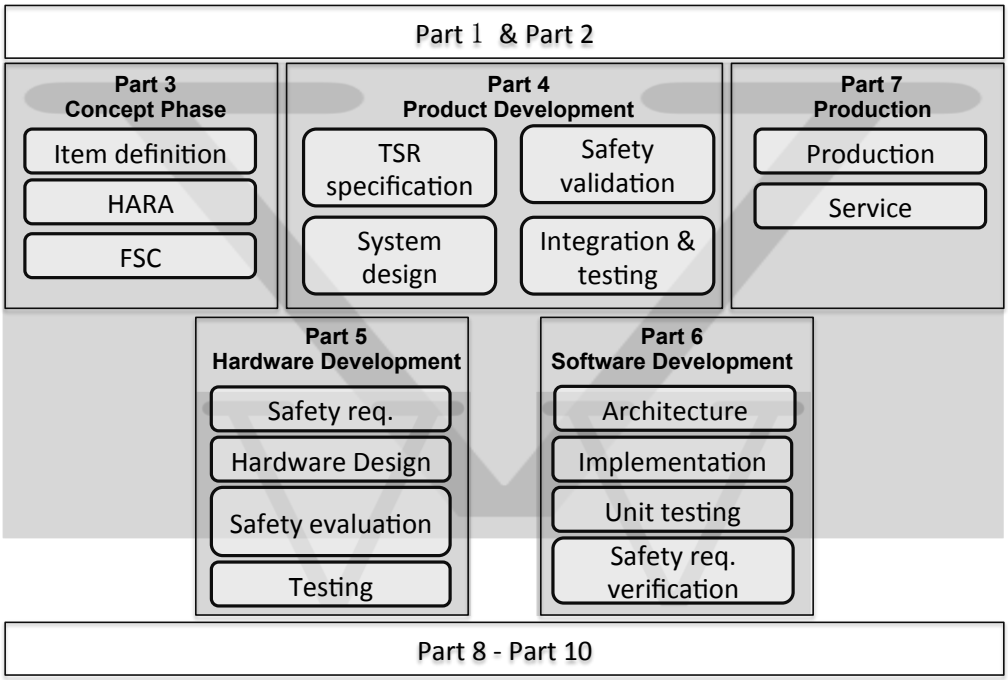


Figure 2.1: An overview of the ISO 26262 V-model.

distinguished: the safety standard level and the project level. At the safety standard level, there are safety standards designed by safety organizations to ensure product safety. At the project level, there are work products as the outputs from each phase of the product development process. The relation between these two levels (Figure 2.2) is the compliance argument: it proves that the requirements of process and product at the project level adopt the recommendations of the ISO 26262 standard. The challenge here is that at the end of the project or at important milestones, evidence must be provided; consequently, a mapping between specific project files and work products in the standard must be made. In summary: there are process and product aspects at the project level, both of which must adhere to the standard.

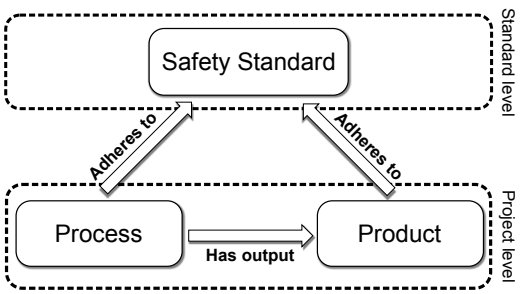


Figure 2.2: Relation between standard and project.

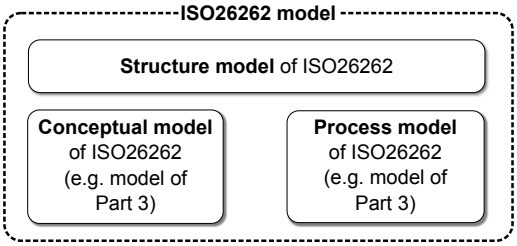


Figure 2.3: Overview of ISO 26262 model.

2.2.2 ISO 26262 Models

Given its attention to both product and process, the safety standard provides a reusable guideline for the development of safety-critical systems; thus, artifacts used in the demonstration of compliance with this reusable guideline can themselves become reusable. However, an enormous amount of manual work is normally involved in interpreting the standard and ensuring compliance with the standard. Therefore, the overall cost of understanding and applying the standard becomes very high. To overcome this drawback, a model-based approach is proposed. The purpose of modeling is to get a better understanding of a safety standard. Compared to the textual standards, models of a safety standard are machine processable. These models provide the predictable, deterministic, repeatable results for validating, which facilitates the reuse of safety assurance data.

In this chapter, three kinds of models are proposed for the safety standards. The structure model and the conceptual model are introduced to support unambiguous understanding of the standard; the process model supports the demonstration of compliance of the process of the project with the process described in the standard. The overview of the resulting model is depicted in Figure 2.3.

Since it is a challenge to describe the whole standard here, we built the ISO 26262 structure model (Figure 2.4) mainly according to the table of contents (see Appendix A). The definitions of some main concepts in the structure model are listed as follows:

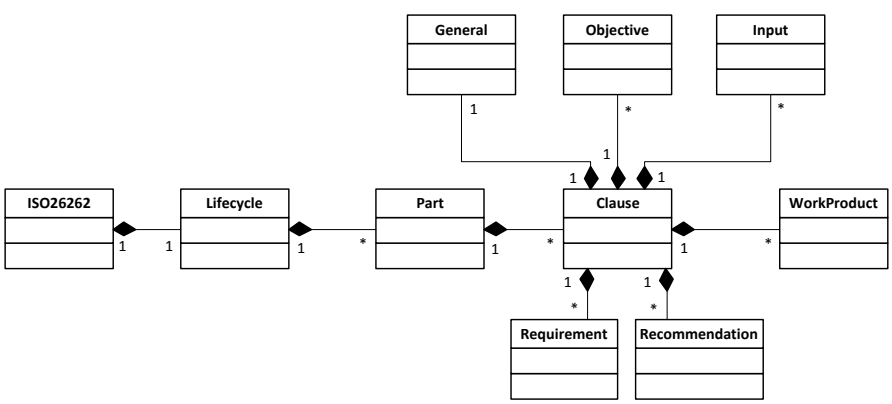


Figure 2.4: Structure model of ISO 26262.

- Lifecycle: *"entirety of phases from concept through decommissioning of the item."* (ISO 26262 Part 1:1.72)
- Phase: *"stage in the safety lifecycle that is specified in a distinct part of ISO 26262."* (ISO 26262 Part 1:1.89)
- Requirements:*"a necessary attribute in a system; a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a user."* [123]
- Work products: *"results of one or more associated requirements of ISO 26262."* (ISO 26262 Part 1:1.142) It means a work product should fulfill one or more requirements of ISO 26262 to demonstrate the compliance.

A safety lifecycle is specified in ISO 26262. Each phase in the safety lifecycle (or each part in the safety standard) includes a number of clauses. In each clause, there is an *Objective* and a *General* section to give a brief introduction of the purpose of each clause. In the *Requirements and Recommendations* section, requirements are discussed in detail. Additionally, the inputs to the clause are listed, while the work products are seen as the outputs.

Due to the different characteristics and aims of ISO 26262 models (structure model, conceptual model, and process model), different methods are chosen to extract and describe these models. Most of the selected description methods in Table 2.1 are widely used in industry.

For the conceptual model, we defined the Snowball approach for extraction (see Section 2.3.1). The results are represented as an Ecore model; then EMF is used for visualization. The Software & Systems Process Engineering Metamodel (SPEM) [114] is used for describing our process model, and the SPEM supporting tool Eclipse Process Framework (EPF) [3] is used. In Section 2.3, the conceptual and process models are explained in detail.

Table 2.1: Methods and tools used for each model.

Model	Purpose	Extraction method	Description method	Tool
Structure Model	showing the structure of the standard	Manual modeling of the table of content	UML [143]	Microsoft Visio 2010
Conceptual Model	capturing the main concepts or terms used in the standard and their relations	Snowball approach	Ecore [2]	EMF [2]
Process Model	demostrating the required process described in the standard	Mapping between standard and SPEM	SPEM [114]	EPF [3]

2.3 Extracting the Conceptual Model from ISO 26262

This section introduces a rule-based approach for extracting the conceptual model of the ISO 26262 standard. Since work products are the results of relevant requirements, the conceptual model extracted from the requirements plays a role for a guideline for complying to the standard. To obtain this kind of model, most current work is based on expert experience; consequently, traceability of the modeling process is ignored in manually creating and maintaining relationships. We have developed the Snowball approach to address this issue.

2.3.1 The Snowball approach

The safety standard contains both high-level requirements, such as those described in *Objective* sections of the ISO 26262 standard, and low-level requirements, represented in *Requirements and Recommendation* sections of the ISO 26262 standard. The Snowball approach aims to assist users of the standard in generating a conceptual model of the standard.

Just like creating the snowman, it involves four steps as follows (see Figure 2.5): First, a basic model with a number of concepts is created from the high-level requirements. Second, like rolling a snowball in the snow, the size of basic model becomes bigger and bigger when processing the low-level requirements according to the rules. Third, the big snowball is shaped into a "snowman" frame, which is the conceptual model of the standard and preliminary model for practical use. Finally, the snowman frame turns to a real snowman after being validated by domain experts.

Therefore, for reducing the subjectiveness of the process of extracting conceptual model from ISO 26262, some rules and steps are prerequisites.

2.3.1.1 Rules in the Snowball approach

The conceptual model of the standard consists of a number of concepts and relations. Therefore, as part of the Snowball approach, rules are defined for selecting those concepts and relations. This not only ensures consistency with the standard, but also supports traceability.

The rules for selecting concepts are as follows:

1. Most concepts arising from the terminology in the standard are originally assumed to be safety related. We then define several categories for grouping the concepts according to their (safety-related) purpose, such as process, product, error, etc. Those categories can be used for modeling other safety standards. Those concepts that do not belong to any category are classified as non-safety related. The remaining concepts are classified as safety related, and are then used in the Snowball approach.

2. Clause titles are not classified as concepts in the conceptual model. Rather, they represent activities in the safety lifecycle, which will be included in the process model.

3. Safety related nouns in the high-level requirements (objectives) are selected as concepts, such as *Item*, *Functional Safety Requirement*, etc.

4. Safety and non-safety related nouns in low-level requirements that have a clear relationship with the existing concepts are selected as concepts, such as *Functional Requirement*, *Non-Functional Requirement*, etc.

The rules for selecting relations are as follows:

1. Verbs between concepts are selected as relations, such as “has”, “is determined by”, “is resulted in”, “is based on”, etc.

2. If there is no specific verb between concepts, the relations are defined according to the prepositions; for instance, in “the boundary of item”, the relation will be treated as “Item has boundary”.

The rules for refinement and optimization of the conceptual model are as follows:

1. All concepts that are instances of more general concepts need to be grouped. Some concepts can be seen as examples of more general concepts. For instance, *Functional Requirement*, *Non-Functional Requirement*, *Functional Safety Requirement* can be grouped into a single concept named *Requirement*.

2. Synonyms should be merged. In ISO 26262, seemingly different concepts sometimes represent a similar idea. For instance, *Item* and *System*, *Software Component* and *Software Unit*, *Safety Lifecycle Activity* and *Safety Activity*. Based on their respective definitions in the standard glossary, those concepts are treated as synonyms.

3. Relations must be precisely defined. Every relation in the conceptual model must have a unique name, and each relation must be clearly defined. For instance, when defining the relation as “has” between *Hazard* and *ASIL*, a following word is given, like “hasASIL”. It could be used to specify what this relation is.

4. If a concept, which is not included in the terminology, only has one connection with other concepts and this connection indicates an ownership relation, then the concept is changed to the attribute of other concepts, for example, concept *Boundary* is only connected with concept *Item*, and their relation indicates that an item has boundary. Thus, the concept *Boundary* is changed to an attribute of the concept *Item*.

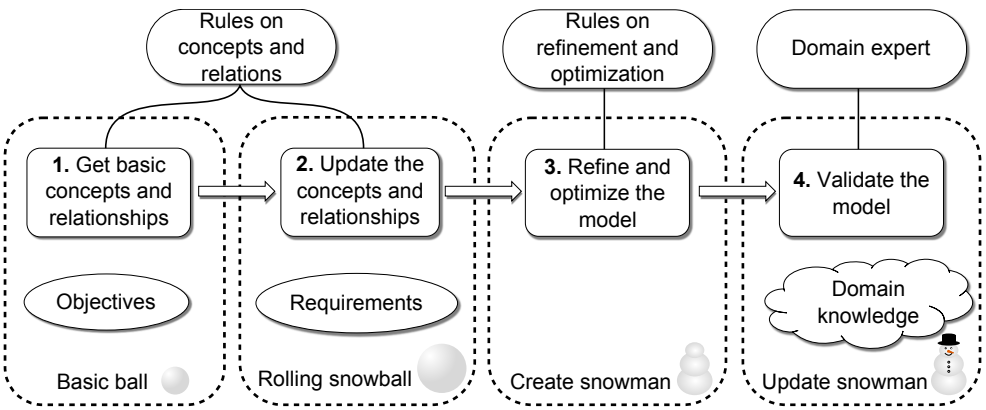


Figure 2.5: Steps in Snowball approach.

2.3.1.2 Steps in the Snowball approach

In a concrete application of the approach, ISO 26262 Part 3 (the Concept Phase) was carefully analyzed manually, which involved both the high level and low level requirements. As discussed earlier, there are four main steps in the Snowball approach (Figure 2.5). In each step, rules for selecting concepts and relations are implemented. The following illustration focuses on the *Item Definition* clause in Part 3.

- Create the basic ball
  We start from the objectives in ISO 26262 Part 3 to identify the basic concepts and relations. In this step, rules on concepts and relations are used. The result of this step is seen as the basic ball in the Snowball approach. The basic ball of *Item Definition* is shown in Figure 2.6, where the classes represent concepts and the association links represent the relations between concepts.

  The original basic ball for the ISO 26262 Part 3 consists of 14 concepts. For each concept, an annotation is added to record the section number of its location in

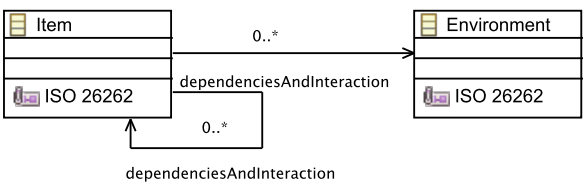


Figure 2.6: Basic ball of *Item Definition*: the annotation of each concept stores the section numbers of its location in the ISO 26262.

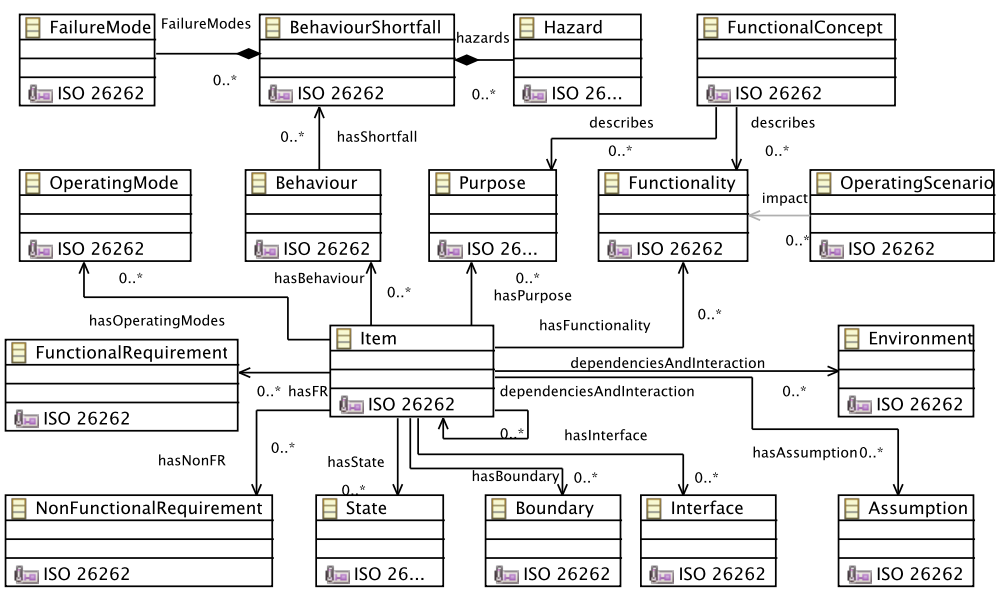


Figure 2.7: Big ball of *Item Definition*.

standard. In this way, the traceability of the modeling process is ensured; it also provides support to the user for modification.

- Rolling the ball
  A careful analysis of the detailed requirements in ISO 26262 Part 3 led us to add more concepts and relations to the basic ball. As in the first step, rules on concepts and relations are used. For example, from "the functional and non-functional requirements of the item", we could obtain two new concepts related with the existing concept *Item*: *Functional Requirement* and *Non-functional Requirement*. The result of this step is seen as the big ball in the Snowball approach. A fragment of the big ball of *Item Definition* is shown in Figure 2.7. The original big ball for Part 3 consists of 59 concepts.

- Create a snowman based on the big ball
  Then, a refinement of the big ball is performed according to the rules on refinement and optimization. The result of this step becomes the conceptual model of the standard. Figure 2.8 shows a fragment of the conceptual model of *Item Definition* phase. The original conceptual model for Part 3 consists of 26 concepts. Many concepts of the big ball are grouped according to the refinement and optimization rule. For example, the concepts *State*, *Boundary*, *Interface*, *Assumption*, *Environment* are grouped as attributes of the concept *Item*.

- Update the snowman
  Finally, domain knowledge is a prerequisite to transform the rough basis of the snowman into the real snowman. In other words, the conceptual model needs to be validated by domain experts before being used in practice. The details of this step are explained in Section 2.5.

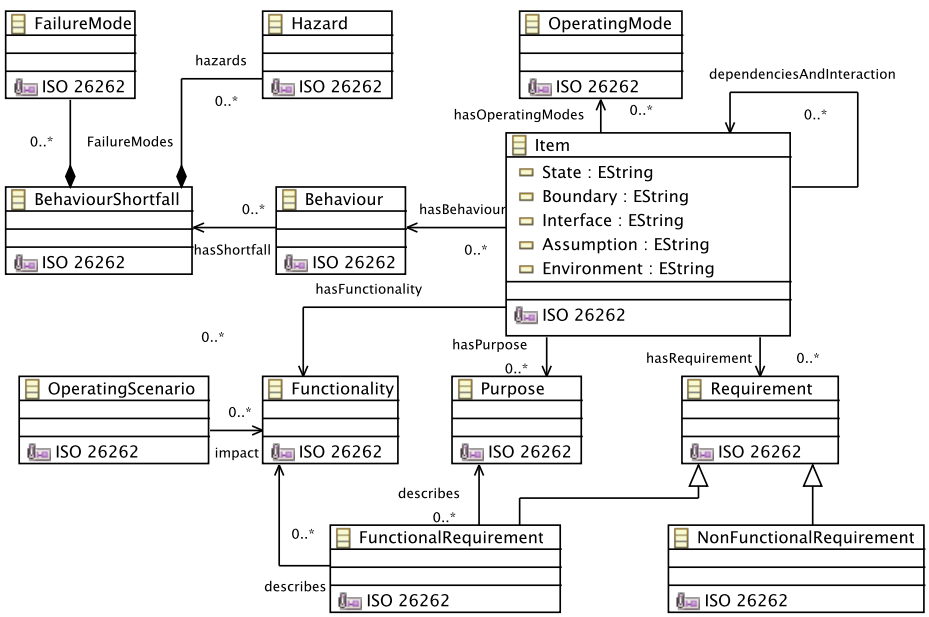


Figure 2.8: Conceptual model of *Item Definition*: the information of synonyms is also stored in the annotations.

Table 2.2: Mapping between concepts of ISO 26262 and SPEM.

Concepts in ISO 26262	Concepts in SPEM
Development process	Process
Part	Phase
Clause	Activity
Objective	Purpose
Requirement and Recommendation	Task
General/Content of requirement	Description
Work Product	Work Product
Note	Guideline

2.4 Extracting Process Model From ISO 26262

2.4.1 ISO 26262 process model and SPEM

As mentioned in Section 2.2, ISO 26262 is based upon a V-Model as a reference process model for the different phases of product development. Therefore, as mentioned earlier, it is claimed to be effectively a reusable guideline for practical domain applications. We selected SPEM for describing the process. The SPEM is defined as a meta-model as well as a UML profile by Object Management Group (OMG) for process modeling.

Since the process described in the standard is more specific, a mapping from the standard to SPEM is needed. The concept mapping is defined in Table 2.2. Compared with the mapping in [31], our aim is to describe a more general process model; therefore, the mapping only focuses on high-level concepts.

2.4.2 Process Model of ISO26262 Part 3

In the following, we use the Eclipse Process Framework (EPF) Composer to build the process model of ISO 26262 Part 3. The EPF Composer uses SPEM version 2.0, and enables users to create processes with breakdown structure editors and workflow diagrams through use of multi-presentation process editors, different process views and synchronization of all views with process changes [3]. From the work breakdown structure view (shown in Figure 2.9), we can see that Part 3 is defined as a phase, whereby each clause in Part 3 is defined as an activity. After an analysis of the requirements, the tasks and their steps as well as the work products are defined for activities.

For the *Item Definition* clause in ISO 26262, there are two main requirements. The definition of tasks in this clause depends on the content of those requirements. In Figure 2.9, there are two tasks defined for this clause. For other clauses, the requirements are represented in groups, such as in the *Initiation of Safety Lifecycle* clause, where there are two requirement groups. In the workflow diagram (Figure 2.10) we can see that two tasks are defined. Further analysis revealed that those detailed requirements could be categorized into two groups. Based on this, two steps have been defined for the *Impact analysis and possible safety lifecycle tailoring* task. Therefore, the activities, tasks, and steps in the process model are defined according to the requirements in different levels. Finally, the work products are allocated to the relevant activities, tasks, or steps. They are all assigned to specific roles, such as safety project manager. In Figure 2.11, the detailed activity diagram shows this relation.

ISO26262 Part3

Presentation Name	Index	P...	M	Type
ISO26262	0			Delivery Process
Part3 concept phase	1			Phase
Item definition	2			Activity
Functional and non-functional requirements identification	3			Task Descriptor
Item boundary definition	4	3		Task Descriptor
Safety lifecycle initiation	5	2		Activity
Development category determination	6			Task Descriptor
Impact analysis and possible safety lifecycle tailoring	7			Task Descriptor
Hazard analysis and risk assessment	8	5		Activity
Hazard analysis and risk assessment initiation	9			Task Descriptor
Situation Analysis and Hazard Identification	10	9		Task Descriptor
Classification of hazardous events	11	10		Task Descriptor
Determination of ASIL and safety goals	12	11		Task Descriptor
Verification Task	13	12		Task Descriptor
Functional safety concept definition	14	8		Activity
Derivation of functional safety requirements	15			Task Descriptor
Allocation of functional safety requirements	16	15		Task Descriptor
Validation criteria	17	16		Task Descriptor

Description | Work Breakdown Structure | Team Allocation | Work Product Usage | Consolidated View

Figure 2.9: Work Breakdown Structure view of the ISO 26262 Part 3 process model: only the name of the phases, activites, and tasks are shown in this figure.

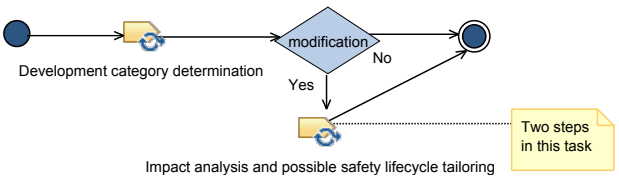


Figure 2.10: Activity diagrams for safety lifecycle initiation task: Safety lifecycle initiation activity diagram.

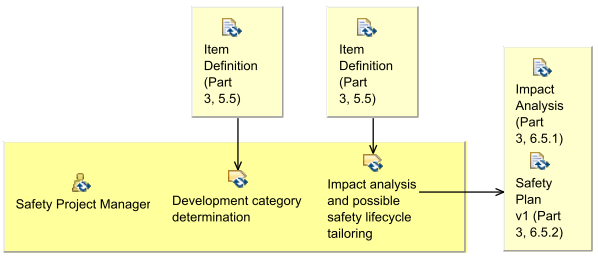


Figure 2.11: Activity diagrams for safety lifecycle initiation task: Safety lifecycle initiation activity detail diagram.

2.5 Model validation

Validation focuses on the conceptual and process models. The last step of Figure 2.5 depicts the validation of the conceptual model. As noted earlier, an important goal of the Snowball approach is to achieve objectivity and cost-effectiveness in the production of the conceptual model, by automating as much as possible. But to be usable, the model must be valid: in other words, it must become "a real snowman."

In a preliminary validation activity, in the context of the OPENCOSS project, we compared our model with the industrial models that are created by the project partners

and used for compliance with ISO 26262 Part 3. Over 90% of the concepts in those models are covered by our conceptual model. Because ISO 26262 describes generic requirements and technologies for functional safety in the automotive domain, different companies could have different interpretations on the standard. Due to their specific focus, some concepts in industrial models have often been renamed; however, the corresponding concepts in the standard can still be identified. For example, in one case "Malfunction" was changed to "Malfunctioning behavior"; in another, "Preliminary architectural assumption" was changed to "Preliminary functional and architectural assumptions"; and so forth. Thus, we see that in the implementation of a model, some concepts are modified according to the specific project context; importantly, however, traceability is maintained by recording the section number of the concepts' locations in standard, so that the source of the concepts and relations can be traced.

The process model is almost identical for applying the standard, except for the special case of the Safety Element Out Of Context (SEooC) [12]. "A Safety Element out of Context (SEooC) is a safety element for which an item does not exist at the time of the development. A SEooC can either be a subsystem, a software component, or a hardware component." (ISO 26262 Part 10:10.1) Note that, *Item* in the ISO 26262 is for example the car itself. For the SEooC, the initiation of safety lifecycle activity is skipped and assumptions are made about the *Concept Phase*, resulting in a truncated process.

As encouraging as this result was, it is also necessary for the conceptual model to be validated by domain experts, because semantic alterations might have been introduced in the conceptual model due to the semi-formal nature of the techniques used. For this purpose, we had the conceptual model examined by two experts (senior consultant and functional safety manager) who have deep experience with the teaching and application of the ISO 26262 standard in industrial projects (both of them over 12 years working practice in functional safety and security). They found some cases of subtle misinterpretation, such as the identification of Item and System as strictly synonymous in the preliminary conceptual model. In real-world industrial application of the standard, an Item is nearly always a System, but in some (rare) cases it can be a Function (e.g. when new functionality is introduced on the same hardware platform).

In addition to concept classes and names, the relations between them are examined and validated by the domain experts. Their explicit representation makes possible inconsistencies emerge quickly. For example, the cardinality of a relationship between the Safety Goal and Safe State was not clear to the domain experts: was it 1:1? This would be wrong, because in practice there may be *no* safe state in automotive scenarios. In another example, both the Hazardous Event and Safety Goal concepts had a relationship with an ASIL concept. In practice, however, it must be ensured that it is always the *same* ASIL, and this must be reflected in the model.

Such subtleties can be difficult or impossible for the rule based techniques of the Snowball approach to identify, partly because they often do not even emerge out of the standard itself but rather its real-world application. For this reason, validation by domain experts is likely to remain a necessary final step even after significant improvement in the techniques employed in the Snowball approach. Nevertheless, experience with domain experts confirmed that a formal representation of the conceptual model is an excellent basis for validation and the eventual elimination of inconsistencies. Furthermore, the availability of the rule based techniques enables non-experts to build a conceptual model, then it can save up to 80% the amount of time that needs to be invested by domain experts.

Once the general model is validated, it is suitable for practical use in the context

of compliance demonstration: Recalling that compliance is demonstrated through the relationship between the standard's model and the project's model, and that traceability is provided at both standard and project level, the effects of modifications at project level can now be traced to the standard level and the impact on the assurance argument determined and those assurance artifacts reused that are not affected by the modifications.

2.6 Related Work

As mentioned in the Section 2.1, some research can be found on modeling safety standards. A conceptual model for the aeronautic standard DO 178B is created to improve communication and collaboration among safety engineers and software engineers [153]. A conceptual model of the generic standard IEC 61508 for electrical and electronic equipment is proposed for the development of compliant embedded software [123].

Other work has advocated model based approaches to enabling assurance reuse. An approach was proposed for evolutionary chains of evidence to support evidence reuse [54]. With a specific focus on evidence, Vara et al. describe the benefits of using model-driven engineering (MDE) to support compliance with safety standards. They further argue that MDE could support the interpretation of standards; specialization of standards to industrial contexts; alignment of standards to organizational practices; planning for certification, etc. Besides these benefits, they argue that MDE could also be used for supporting safety assurance reuse management, such as evidence reuse, safety case reuse etc. Some research can be found on safety case reuse [128] [118]. Goal Structuring Notation (GSN) is introduced as a graphical modeling approach for safety case construction [87]. Then using the pattern and modular extensions of the GSN, Palin et al. propose a number of reusable safety arguments which covers all parts of ISO 26262 and the issues of compliance and assurance [118]. Moreover, a Case-Based Reasoning (CBR) approach has been proposed for modeling, retrieving and reusing previously assured safety cases [128].

2.7 Conclusions

In this chapter, we presented a model-based approach to enabling safety assurance reuse through objective and cost-efficient modeling of the relevant standards. The Snowball methodology provides rules for extracting the conceptual model from a safety standard, thereby reducing the amount of manual work involved. Over 90% of the concepts in the industrial models are covered by our conceptual model. A better result will be obtained if the domain experts are involved in all the steps of Snowball approach, but it will be more costly. The availability of the rule based techniques enables non-experts to build a conceptual model, then it can save up to 80% the amount of time that needs to be invested by domain experts. The process in the standard is modeled with the OMG SPEM. Although the approach currently operates at a very high level, it provides a basis for describing a process model in the context of the safety standards. Moreover, the conceptual models of safety standards can be also used as inputs in Chapter 3 and Chapter 5. The process model for the ISO 26262 can also facilitate the approach described in Chapter 7.

Chapter 3

Metamodeling for Safety Assurance

In this chapter, we propose to use conceptual models in the form of metamodels to support certification data reuse and facilitate safety compliance. A metamodel transformation approach is outlined to derive domain or project specific metamodels using a generic metamodel as basis. Furthermore, we present a metamodel refinement language, which is a domain-specific language that facilitates simple refinement of metamodels. Finally, we use two case studies from the automotive domain to demonstrate our approach.

3.1 Introduction

Model-driven engineering has been introduced to increase the level of abstraction at which we reason about software, and reduce software development cost [144]. Recently, it has also been used to reduce the high costs for safety assurance. Modeling safety standards facilitates compliance demonstration and safety assessment [123]. Analyzing current safety standards and safety argumentation provides an efficient way of how to reuse current certification data (see Chapter 5). Additionally, model-driven technology supports system suppliers in managing the system development process [78] as well as the evidence for safety assurance generated from that process [122].

A metamodel used in safety-critical domains can be a conceptual model with concept-mediated semantics [124]. *"Conceptual modelling aims to create an abstract representation of the situation under investigation, or more precisely, the way users think about it. Conceptual models enhance models with concepts that are commonly shared within a community or at least between the stakeholders involved in the modelling process."* [141] Concept-mediated-semantics means the elements in a conceptual model represent concepts, and these concepts represent entities in the domain. For instance, the elements of the aforementioned metamodels represent safety related concepts, which represent entities ("the real world") in safety-critical domains. Different conceptual models are proposed or created based on different usages. In recent years, some companies have started investigating how to build their own conceptual models based on their domain knowledge. Consequently, a number of conceptual models have been built, for example an IEC 61508

conceptual model [123], and an ISO 26262 conceptual model (See Chapter 2). Those conceptual models are called domain-specific conceptual models. Each company might use some different concepts and relations to describe safety certification data. Therefore, the certification data or expertise reuse between different companies or domains tends to be a big challenge.

As discussed in Chapter 1, a Generic MetaModel (GMM) of safety standards was developed in the OPENCOSS project. The benefits of the GMM are: patterns of certification assessment can be shared, and cost-effective re-certification between different standards is supported [145] [19]. Consequently, when introducing the GMM to the companies in safety-critical domains, their current way of working must be changed to conform to the GMM, and extra cost will be involved. In practice, those companies want to get the benefits of the GMM while minimizing the required changes to do so. Additionally, because the concepts in the GMM are limited and generic, some ambiguities may rise when interpreting and using those concepts.

In this chapter, we propose an approach to reduce the loss of descriptive capacity and keep or even increase the benefits. The approach uses the GMM as starting point, and introduces a specific conceptual model in the form of Specific MetaModel (SMM) for specific usage. Model transformation technology can be used to bridge the gap between the conceptual models at different abstraction levels [119]. We present a metamodel transformation approach to facilitate the process of creating metamodels for a specific safety standard, domain or company (Section 3.2). Also, our approach supports reuse by an automatic comparison between concepts in different domains using the traceable changes that are documented in the metamodel transformations. A MetaModel Refinement Language (MMRL) is defined to support system suppliers to build their own metamodels based on the GMM. The language allows system suppliers to introduce domain or project concepts to the GMM, and then to generate a specific model editor based on the resulting SMM. The system suppliers could use this editor to built their models, such as standard models or process models, which conform to their SMM. Moreover, the traceability information can be found in MMRL specifications and analyzed for conceptual mappings between different metamodels (derived from the same metamodel). Therefore, it facilitates the certification data reuse between different companies or domains.

The remaining chapter is organized as follows: Section 3.2 outlines our overall metamodel transformation approach. Section 3.3 presents a domain specific language for refining metamodels. Section 3.4 discusses the results of two case studies from the automotive domain. Section 3.5 introduces the related work of our approach. Finally, Section 3.6 summarizes this chapter.

3.2 Metamodel Transformation

As mentioned before, the current generic metamodel (GMM) proposal oversimplifies the modeling needs of safety engineers and assessors. This can lead to overgeneralization, additional manual work, and less support for automatic consistency checks. To address this, extended metamodels are refined from the GMM; if new domain concepts are required for the system supplier's purpose, they will be added into the GMM. Extending the GMM into domain-specific metamodels provides the advantage of creating a language that is closer to the world of expertise of the safety engineer in a certain domain and will ease the effort to produce the reference assurance. Besides, domain-specific metamodels can reduce the ambiguities in the GMM to prevent safety engineers from making interpretation

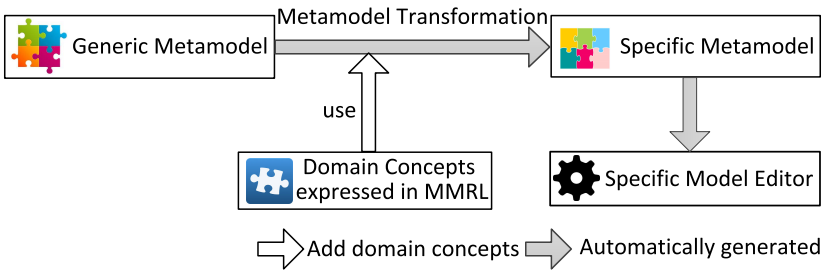


Figure 3.1: Overview of our approach.

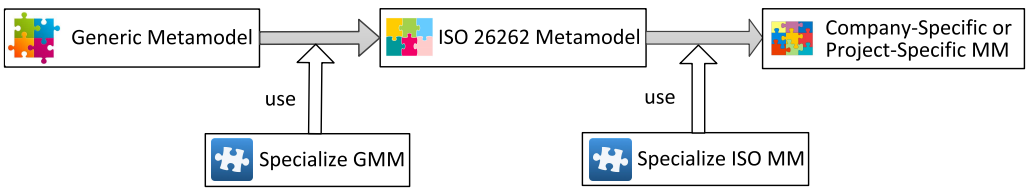


Figure 3.2: Generic use of our approach.

mistakes while creating models. Additionally, domain-specific metamodels only need to be defined once by the best expert(s) available in a certain domain.

An overview of our approach is shown in Figure 3.1. We define a metamodel refinement language (MMRL) to introduce domain concepts into the GMM. A metamodel transformation is executed, which takes the GMM along with those domain concepts described in MMRL as inputs, and produces an SMM as output. Finally, a graphical editor, based on the SMM, can be automatically generated, which enables safety engineers to build their models using concepts from their own domain.

Our approach can be divided into multiple specialization steps. Figure 3.2 shows a two-steps scenario. In the first step, the GMM is updated by adding domain concepts. For example, by adding concepts from the ISO 26262 standard, an ISO 26262 metamodel can be obtained. Then, in the second step, the company-specific or project-specific metamodels can be generated from the updated GMM. For instance, for a company X, a project metamodel for fuel cars and a project metamodel for electrical cars can be derived from the same ISO 26262 metamodel. Then by analyzing the corresponding MMRL specifications, a conceptual mapping between those two metamodels are found through the ISO 26262 metamodel. Finally, reuse of safety-assurance data between these two projects can be supported based on the conceptual mapping. Therefore, the two key benefits of our approach are that the changes made to the GMM are documented and defined unambiguously in the form of refinement specifications, thus making traceability amenable for analysis, and the SMMs or the intermediate domain-specific metamodels can be reused in other domains or by other companies.

3.3 Metamodel Refinement Language

We defined and implemented a MetaModel Refinement Language (MMRL) that can be used to refine metamodels. It is a simple domain-specific language, which offers operations to rename elements of a metamodel, replace annotations of metamodel elements, make

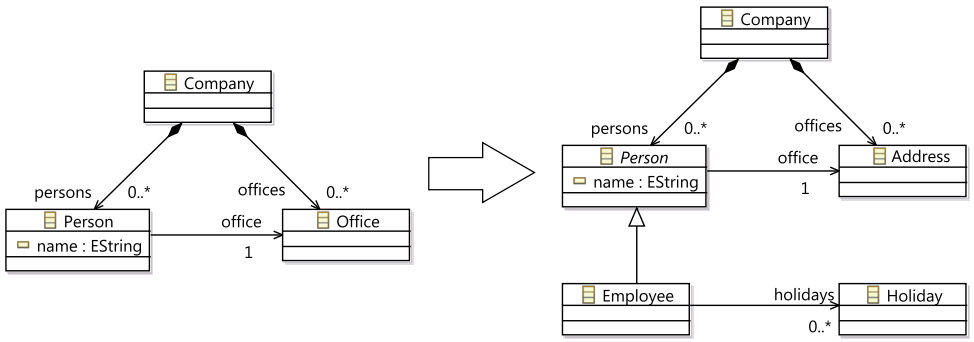


Figure 3.3: The metamodel on the right is transformed into the metamodel on the left by applying the refinement specification in Listing 3.1. These two metamodels reside in a package *company*.

certain metaclasses of a metamodel abstract, and add elements to a metamodel. By applying these operations to an existing metamodel, a new, refined version of this metamodel is created. Each refinement specification defined using our MMRL clearly specifies the relation between the input and output metamodel. The syntax of the language is defined with EMFText[1], and the language has been implemented using the Epsilon Transformation Language (ETL) [131].

The metamodel refinement language allows the system supplier to describe their domain concepts in terms of the existing GMM concepts using the provided operators.

The operations can be classified as follows:

- The structural feature operations *AddPackage*, *AddClass*, *AddAttribute*, *AddDataType* and *AddReference* are defined for adding structural features of metamodels.
- The Annotation operations *ReplaceAnnotation* and *AddAnnotation* are defined to add annotations of metamodels, which can support generating graphical editors from metamodels.
- The enumeration operations: *AddEnum* and *AddEnumLiteral* are used to create a new enumeration class or a new literal for existing enumeration class.
- The modification operations: *Abstract* and *RenameElement* enable system suppliers to rename some elements (package, class, attribute and reference etc) or make some classes abstract. If a class becomes abstract, it means that class will not be visible in a graphical editor.

In our MMRL, there is no operation for deleting elements from the input metamodel. In our approach, the input is the GMM, but it is not GMM specific. All the concepts in the GMM have been selected and validated by domain experts. Therefore, we assume that there are no redundant elements in the GMM. If a concept in the GMM does not exist in some domain, they will be changed to an abstract class instead of removing it. In the following two sections, we discuss structural feature operations and annotation operations, which are the two core parts in the definition of our MMRL. Structural feature operations are used to manipulate the main structure of refined metamodels, and annotation operations facilitate tool generation.

[1] `http://www.emftext.org/`

Listing 3.1: This refinement specification performs five operations: it defines that the class named *Person* in package *company* should be abstract (line 2), it renames the class named *Office* in the same package to *Address* (line 3), it adds a class named *Employee* that inherits from *Person* (line 4) and a class named *Holiday* to this package (line 5), and it adds a reference named *holidays* with type *Holiday*, lower bound 0, and upper bound -1 to the class *Employee* (line 6-8).

```
1 operations
2   abstract company.Person
3   rename company.Office to Address
4   add class Employee {superTypes {company.Person}} to company
5   add class Holiday {} to company
6   add reference holidays {
7     lowerBound 0 upperBound -1 type company.Holiday
8   } to company.Employee
```

Figure 3.3 and Listing 3.1 present a small example to illustrate the process of refining metamodels. On the left of Figure 3.3, a small metamodel is shown, which consists of three classes (*Company*, *Person*, and *Office*) and three references (*persons*, *offices*, and *office*). After applying the refinement specification given in Listing 3.1, this metamodel is transformed into the metamodel shown on the right of Figure 3.3.

3.3.1 Structural Feature Operations

The structural feature operations in MMRL are designed in accordance with the Ecore metamodel [2]. Figure 3.4 shows an extract of our MMRL on the structural feature operations, highlighting the key relationships between four operations: *AddPackage*, *AddClass*, *AddAttribute*, and *AddReference*. An *AddPackage* operation has a *PackageDefinition*. Each *PackageDefinition* has a number of *ClassifierDefinition*s, which are specialized into *ClassDefinition*. As each *AddClass* operation has a *ClassDefinition*, it specifies that a number of new classes could be added to a new package or an existing one. Similarly, a *ClassDefinition* has a number of *StructuralFeatureDefinition*. Then *StructuralFeatureDefinition* is specialized into *ReferenceDefinition* and *AttributeDefinition*, which are associated with *AddReference* and *AddAttribute* operation respectively. Therefore, it shows that a number of references or attributes can be added to a new class or an existing one.

After adding the concrete syntax to our MMRL, a system supplier can call those operations though keywords. For example, in List 3.1, by using a keyword "add class", the *AddClass* operation has been performed. Eventually, two new classes (class *Employee* and class *Holiday*) have been added to the original model.

3.3.2 Annotation Operations

In our MMRL, annotation operations are designed for annotating metamodels during metamodel transformation. Figure 3.5 shows a depiction of the annotation definition in our MMRL. The *ModelElementDefinition* element is the container of *AnnotationDefinition*, while *AnnotationDefinition* is a kind of *ModelElementDefinition*. Each *AnnotationDefinition* has a number of *KeyValuePair*s, where key and value attributes are defined. Two operators use the annotation definition: *ReplaceAnnotation* and *AddAnnotation*. Each of these two operators is associated with a concrete *AnnotationDefinition* and is used for

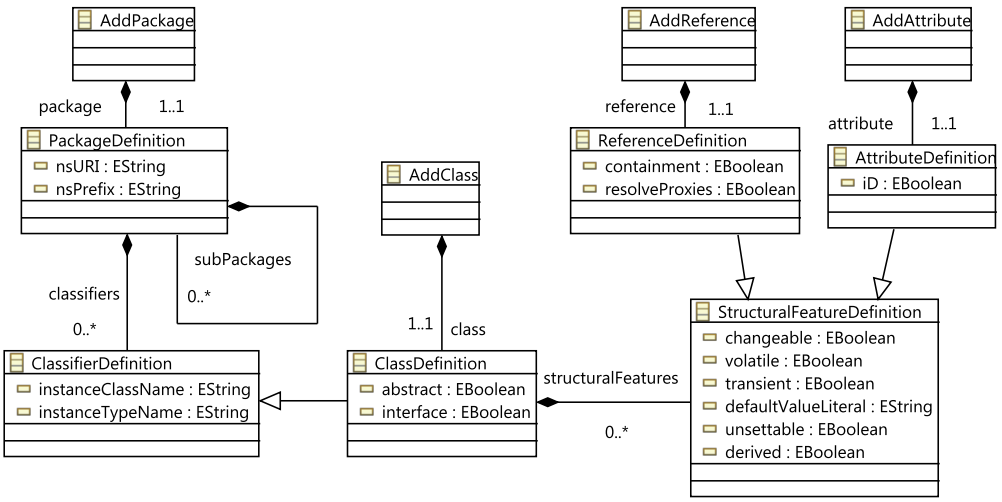


Figure 3.4: Structural feature definition in our MMRL metamodel.

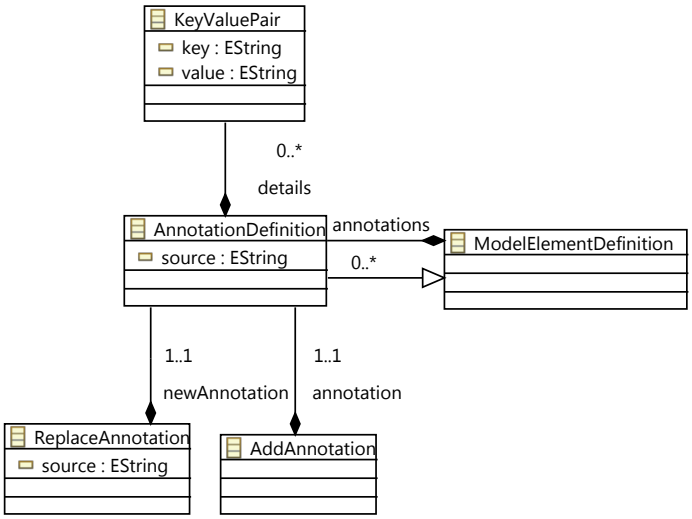


Figure 3.5: Annotation definition in our MMRL metamodel.

modifying current annotations or adding a new one.

As mentioned before, the GMM can be transformed to different SMMs according to system supplier's needs. Then those SMMs can be used to describe the supplier's current way of working in a model. Finally, a user friendly framework can be built based on each specific metamodel. By using this framework, users can create their models based on the metamodel with a clear view of the model structure, which also helps them to identify if some elements are missing or incorrect.

EuGENia is a tool that can generate graphical editors from annotated metamodels [132]. It simplifies the complex graphical editor generating process of GMF and provides a

Listing 3.2: This refinement specification adds an annotation to the class *Person* in the package *company* that can be processed by EuGENia. Among other things, it specifies that this class should be represented as a node (line 1) with a certain label (line 2), size (line 3) and figure (line 4) in the graphical editor.

```
1 add annotation "gmf.node" {
2   "label" = "name",
3   "size" = "70,50",
4   "figure" = "figures.PersonFigure",
5   "label.icon" = "false",
6   "label.placement" = "external"
7 } to company.Person
```

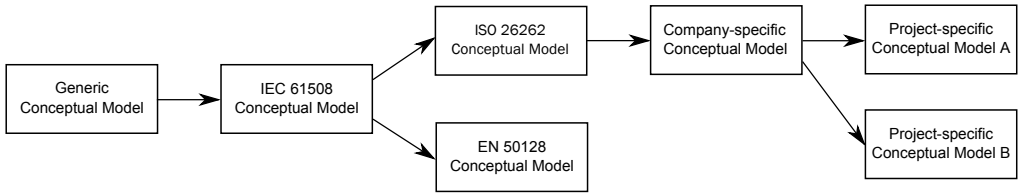


Figure 3.6: By successively refining metamodels, a family of closely related metamodels is created. For clarity, the metamodel transformations and the domain concepts introduced by refinement specifications of Figure 3.1 are not shown in this figure.

number of annotations for users to annotate their metamodels. Because our metamodel refinement language can be used to add annotations to metamodels or replace existing annotations, it can also be used to create metamodels that can be processed by EuGENia directly. The refinement specification in Listing 3.2 provides an example of a specification that adds an annotation for EuGENia.

3.3.3 Sequence of Transformations

By repeatedly applying refinement specifications to metamodels, a family of metamodels can be created (Figure 3.6). By family of metamodels, we refer to a closely related set of metamodels, where each metamodel in the set (except the root metamodel) can be defined in terms of another metamodel from the same set. Each of the metamodels (except the root metamodel) is defined in terms of an existing metamodel and a set of changes. For example, an IEC 61508 metamodel can be refined into an ISO 26262 metamodel and an EN 50128 metamodel with some changes. Some concepts in the original metamodel might not be used in other metamodels. Then these concepts are changed to be abstract instead of being deleted. Therefore, the elements in each of the metamodels can be traced back to a refinement specification (or the corresponding intermediate metamodel) or the metamodel that formed the starting point of the family. For instance, the element *ASIL* in a project-specific conceptual model A can be traced back to the *ASIL* in the company-specific conceptual model, the *ASIL* in the ISO 26262 metamodel, the *SIL* in IEC 61508 metamodel, or the *CriticalityLevel* in the generic conceptual model. As changes of metamodels are stored in metamodel refinement specifications, traceability information in the transformation sequence can be obtained and maintained by analyzing those specifications.

Each metamodel in a family of metamodels could be used as a basis for a tool to edit models that conform to this metamodel. By taking the EuGENia annotations into account while creating a family of metamodels, a closely related set of graphical editors can be generated from this family. The concepts and relations in the models created with these editors can be related across models by tracing their relationships through the refinement specifications.

3.4 Case Study

As mentioned before, we have implemented our approach using the Eclipse Modeling Framework with the following plug-ins: EMFText is used to define the concrete syntax of MMRL, and ETL is used to build the metamodel transformation for metamodel refinement based on the GMM. Finally, a graphical editor can be automatically generated by GMF and EuGENia based on a resulting SMM.

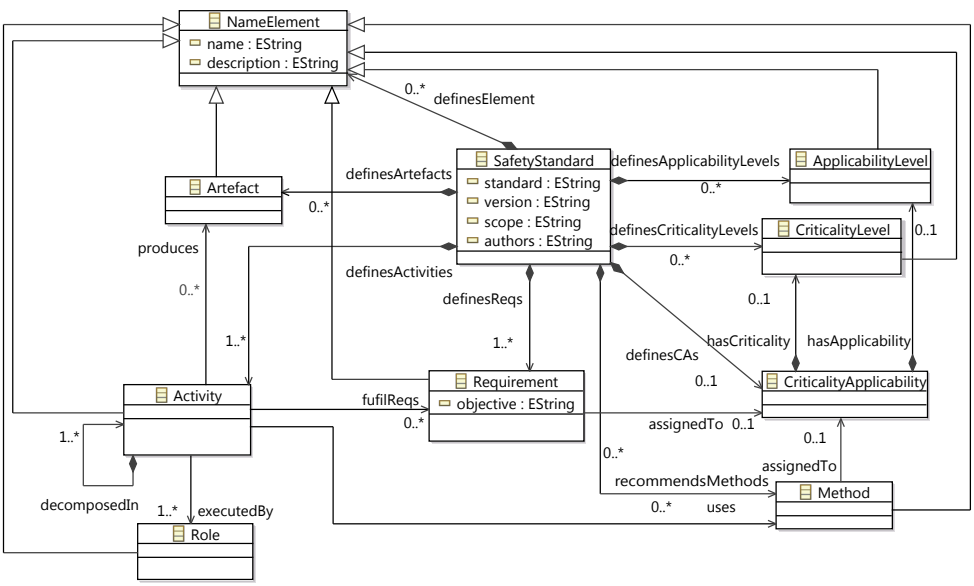


Figure 3.7: The generic metamodel for safety standards.

For our demonstration, we use two case studies from the automotive domain related to ISO 26262[1]: company X and company Y. Figure 3.7 shows an extract of the GMM. The definition of each element in the GMM is presented by Vara et al. [145]. Figure 3.8 shows the refined GMM for two automotive companies according to their needs. This is done by documenting their modification requirements using MMRL operations. The black, light grey, and dark grey elements result from the application of MMRL operations. The black elements are only modified by company X. The light grey elements are only modified by company Y. The dark grey elements are modified by both companies. By comparing the GMM and the refined metamodel, we could see that:

[1]Due to the confidential concerns, we use company X and company Y to represent two different automotive companies.

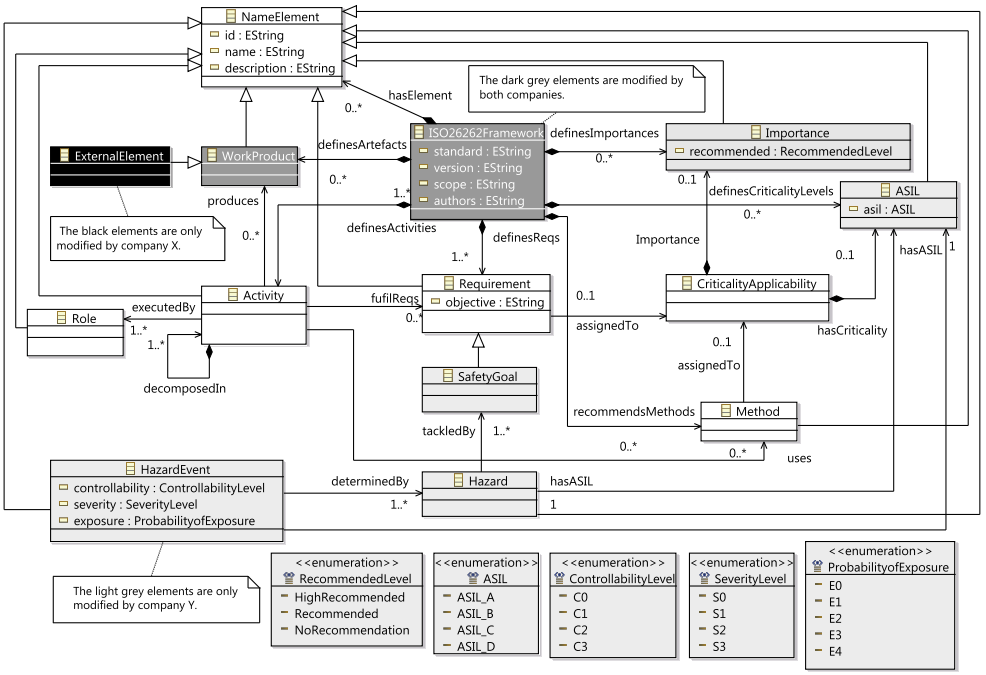


Figure 3.8: Two company-specific metamodels derived from previous GMM.

1. Two classes have been renamed by company X and company Y. *SafetyStandard* and *ArtefactType* have been renamed to *ISO26262Framework* and *WorkProduct* respectively.

2. A new class (*ExternalElement*) has been added by company X.

3. Two classes have been renamed only by company Y. *ApplicabilityLevel* and *CriticalityLevel* have been renamed to *Importance* and *ASIL* respectively.

4. Three new classes have been added by company Y: *SafetyGoal*, *Hazard*, and *HazardEvent*.

5. Seven new references have been added by company Y: one for *SafetyGoal*, three for *Hazard*, and three for *HazardEvent*.

6. Five new enumerations have been introduced by company Y: *RecommandedLevel*, *ASIL*, *ProbabilityofExposure*, *ControllabilityLevel*, and *SeverityLevel*.

7. Five new attributes have been added by company Y: *recommended* for *Important*; *asil* for *ASIL*; and *controllability*, *severity* and *exposure* for *HazardEvent*.

Through the GMM and two refinement specifications, we could see that there are a number of common concepts between the conceptual models of company X and Y. Thus, a conceptual mapping between these two company-specific metamodels can be built. For example, the concept *ExternalElement* in the conceptual model of company X can be mapped to the concept *WorkProduct* in the conceptual model of company Y.

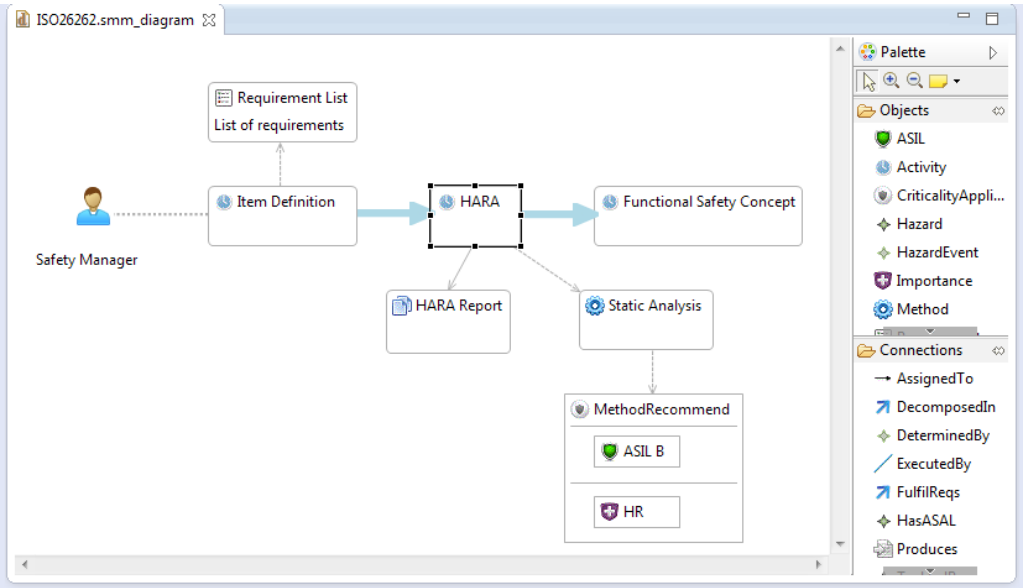


Figure 3.9: A screen shot of a company Y editor.

The concept *Activity* in the conceptual model of company X is the same as the concept *Activity* in the conceptual model of company Y. The conceptual mapping shows which concepts are related in those company-specific metamodels and indicates possible overlaps in safety assurance data. Therefore, this conceptual mapping can be used to support the certification data reuse beween these two companies.

Moreover, in our refinement process, both company-specific metamodels have been annotated. A graphical framework based on each of those metamodels has been generated using EuGENia. Figure 3.9 shows the editor generated for company Y.

Safety engineers of company Y can use the resulting editor to create their own models using ISO 26262 concepts, which helps them to show that their development process complies with the ISO 26262 standard. They can add a new element to their development process model by dragging the relevant element from Objects tool onto the workspace. Besides, they can connect different elements together using the Connections tool. The created models are instances of the company Y metamodel. For example, in Figure 3.9, *Item Definition*, *HARA*, and *Functional Safety Concept* are instances of the class *Activity* in the aforementioned metamodel (Figure 3.8). Besides, they could specify one or more person(s), such as a safety manager, to take charge of those activities. A benefit of our approach is that domain concepts or project-related concepts can be kept, and system suppliers do not need to change their current way of working to conform to the GMM. Thus they can easily learn how to use the generated editor. Besides, the traceability and its documentation from the GMM to the SMM is realized using MMRL.

3.5 Related Work

Related research is found in modeling safety standards, and metamodel refinement.

Modeling safety standards. Conceptual models of safety standards are widely used

in different domains for different usages. As mentioned in Section 6.1, a conceptual model of IEC 61508 is proposed for characterizing the chain of evidence for safety certification in the petroleum industry [123]. A conceptual model of aeronautic standard DO 178B is provided to improve communication and collaboration among safety engineers in the avionic domain [153]. Moreover, conceptual modeling in the context of ISO 26262 has been carried out [21] [90]. In this chapter, we discussed how to generate conceptual models of safety standards from a generic metamodel.

Metamodel refinement. Metamodel refinement is strongly related with metamodel evolution and metamodel adaptation. Wachsmuth describes the use of transformation patterns in the form of QVT relations for metamodel refinement [146]. By introducing new concepts, the target metamodel can be extended though model transformation. A model change language with a number of migration rules is presented by Narayanan et al. for defining metamodel changes [111]. It is a high-level visual language and is designed for describing metamodel evolution. Our approach presented in this chapter is discussed in the context of safety-critical domains and focuses on metamodel refinement with metamodel transformation rather than metamodel evolution. Metamodel evolution is caused by external factors, whereas the metamodel refinement is a design process.

3.6 Conclusions

In this chapter, we proposed a metamodel transformation approach to facilitate safety assurance. Based on previous research, we present a refinement process for a generic metamodel according to a system supplier's input. Then a graphical editor can be generated based on the resulting specific metamodel. Therefore, it not only enables the system supplier to reuse the existing certification data by means of a conceptual mapping, which is supported by the GMM, but also respects their current way of working by means of specific metamodel support. Additionally, a metamodel refinement language is defined. With its help, the traceability information from the GMM to SMMs can be maintained. Refinement specifications support the documentation of the traces between generic and more specific metamodels and vice versa. They can be used to automatically determine the similarity between concepts in different domains. Also, system suppliers can get their own metamodel and tool support easier and quicker. In the next chapter, we will discuss the construction of mappings between specific metamodels based on MMRL transformations, and traceability management and analysis among those transformations.

Chapter 4

Traceability Management and Metamodel Comparison

In safety-critical domains, conceptual models are created in the form of metamodels using different concepts from possibly overlapping domains. Comparison between those conceptual models (or metamodels) can facilitate the reuse of models from one domain to another (See Section 4.4). This chapter describes the mappings detected when comparing metamodels and models used for safety assurance. We use a small use case to discuss the mappings between metamodels and models, and the relations between model elements expressed in mappings. Finally, a case study is provided to demonstrate our approach.

4.1 Introduction

In Chapter 3, we presented a metamodel transformation approach to facilitate the process of creating metamodels for a specific safety standard, domain or company. As a result, a family of metamodels was generated through a metamodel transformation sequence. The traceability information of those metamodels needs to be stored, maintained and analyzed for consistency. Besides, one of the key tasks of reducing the safety assurance cost is to find reusable data from models conforming to similar metamodels. Thus, metamodel comparison and model comparison are vital. The comparison results can be described as mappings between metamodels or between models and used for supporting safety-assurance data reuse.

Our main goal are the results of metamodel comparison and model comparison. We call this kind of mapping comparative mappings. In comparative mappings, the similarities and differences between the source and target model not only depend on metamodel comparison, but also on model comparison. In other words, even if the results of comparison between metamodels are known, the comparison between models still needs to be done manually. However, the former comparison can facilitate the latter one by specializing which type of models must be compared.

In this chapter, we discuss our previous metamodel transformation approach to take into account comparative mapping support. We define two levels of comparative mapping: conceptual mapping and concrete mapping. Conceptual mapping shows the

relations between different conceptual models or metamodels[1]. Concrete mapping shows the relations between models. Then we introduce four comparative mapping types to express the relations between model elements. Finally, an illustrative case study from the OPENCOSS project is used to demonstrate our approach.

4.2 A Small Use Case

To simplify the user scenario and demonstrate our approach in a clear and easy way, we use a small use case from the personnel management domain. The scenario is as follows: there are two companies, company X and company Y. They bought the same personnel management system from a software supplier. This system is developed based on the Eclipse Modeling Framework [2]. Therefore, the data stored in the system are described by metamodels. Then they find they need to improve that system to make it fit to their own situations. Company X would like to use the concept "employee" instead of "person", and introduce a contract into the system. Company Y would like to add only the concept "employee". Therefore the system gets modified by both companies.

Moreover, company X has a branch: branch Z. Branch Z has a lot of developers, which is a special type of employee. Additionally, Branch Z would like to know the age of their employees in their system. Thus, they also modified the company X personnel management system based on their needs.

Later, company X wants to merge company Y into their branch Z. As company Y and branch Z use different systems for managing human resources, branch Z cannot directly import the human resource data from the system of company Y. Branch Z has to manually restructure all the data of company Y and put it into their system. Therefore, the data from company Y is not reusable. However, there are some similarities between the personnel systems of branch Z and company Y. They start to think about finding the similarities and relations to support the data reuse, and to reduce the cost of merging. For instance, for the same person, his or her resume can be reused from company Y system to Branch Z system.

4.2.1 Our Implementation for the Use Case

Figure 4.1 illustrates the sequence of transformations for the aforementioned use case. There is a domain metamodel of the original personnel management system, which represents the relations between *Person* and *Resume*. The domain metamodel is referred to as a generic metamodel of a target domain. Through a metamodel transformation sequence, as described in Chapter 3, it can be refined into the company X metamodel, the branch Z metamodel, and the company Y metamodel. Therefore, in this family of metamodels, there are four metamodels. The traceability information of the refinement process is stored in the metamodels themselves in the form of annotations. For example, from the domain metamodel to the company X metamodel, a new class *Contract* is added and the class *Person* is renamed to *Employee*. In other words, two operations are performed: a rename-element operation and an add-class operation. The details of these two operations are stored in the annotations of *Employee* and *Contract* (Figure 4.1). For the rename-element operation, a *refine.renameElementOperation* annotation is added to *Employee* with a reference to the class *Person* in the domain metamodel. For the

[1] As all conceptual models in this chapter are created in the form of metamodels, in the remainder of this chapter we mention metamodel only.

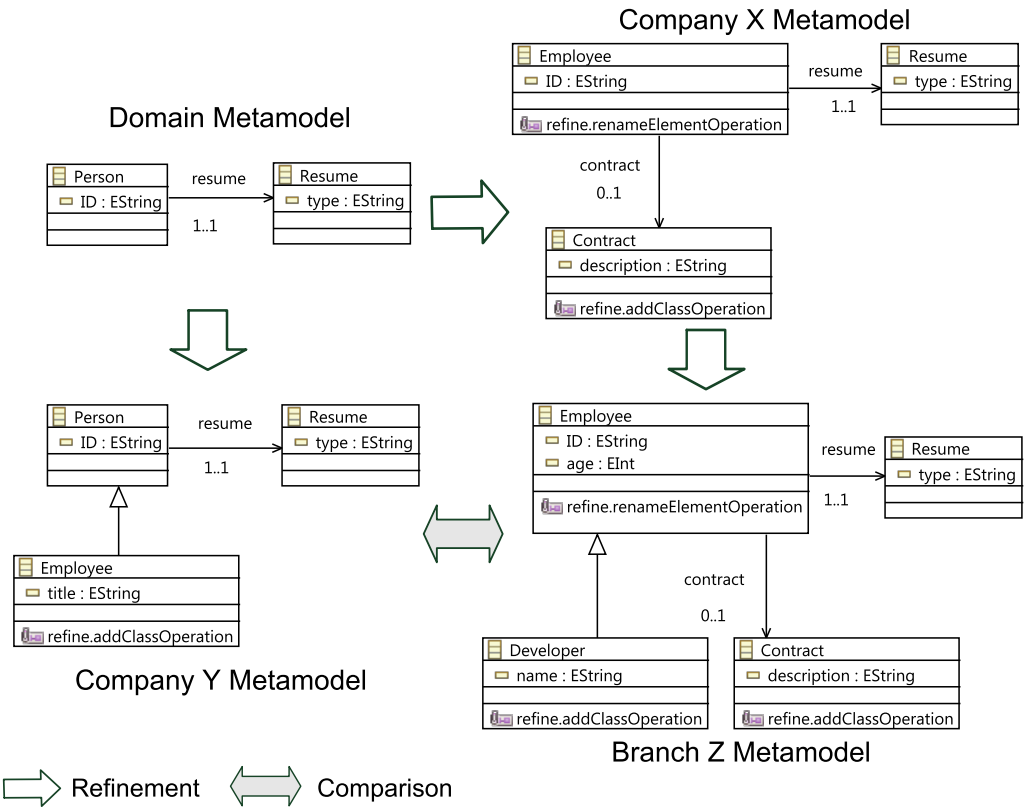


Figure 4.1: A small example for the sequence of transformations.

add-class operation, a *refine.addClassOperation* annotation is added to the class *Contract*. Because the class *Contract* is new, there is no class in the domain metamodel that can be referred to. Thus, the reference of the *refine.addClassOperation* annotation is the class *Contract* itself.

Similarly, from the company X metamodel to the branch Z metamodel, an attribute *age* is added to the existing class *Employee* and *Developer* is introduced as a new type of *Employee*. The information of the corresponding operations is logged in the new annotations of attribute *age* and class *Developer*. The annotation of attribute *age* refers to the class *Employee* in the company X metamodel, while the annotation of the class *Developer* refers to itself. Besides, the old annotations are kept from the company X metamodel.

For each element (concept and relation) in the family of metamodels, its annotation keeps all its traceability information in the transformation sequence. Therefore, the concepts and relations in the metamodels can be related across metamodels by tracing their relationships through their annotations. Furthermore, the traceability information can be obtained and analyzed for comparative mappings between metamodels, which will be discussed in the next section.

4.3 Mapping Support

In this section, we discuss the comparative mapping between metamodels and between their instances. The relation types between concepts and between instances are described in terms of mappings.

4.3.1 Mappings: between metamodels and between models

From a usability perspective, there are two levels of comparative mapping: mappings between concepts and mappings between instances of concepts (on a concrete level). These mappings are illustrated in Figure 4.2 and can be described as follows:

- Conceptual mapping between metamodels: The models in this mapping are all layer 2 MOF models, called M2-models. For example, in Figure 4.1, the company Y metamodel or branch Z metamodel are M2-models [117], therefore, they are also referred as company Y metamodel or the branch Z metamodel. In a reuse scenario, this implies that the conceptual mapping between metamodels only focuses on domain concepts.
- Concrete mapping between models: The models in the mapping are all layer 1 MOF models, called M1-models. For example, company Y models or branch Z models are M1-models.

The conceptual mappings can be implemented through metamodel transformations. In Section 4.2, a refinement of the Domain Metamodel (DMM) is described. The refinement specifications involve the changes made from the DMM to a targeted metamodel. The corresponding conceptual mapping from the DMM to a targeted metamodel is automatically documented in the metamodel transformation. For example, Listing 4.1 shows the mapping information from the domain metamodel to the company X metamodel in the form of XML. The comparisons between different metamodels (except the DMM) can be obtained by analyzing the mapping documents of each metamodel. We designed a script file based on Epsilon Object Language [130] to achieve this automatically. Listing 4.2 shows the comparison results between the company Y metamodel and the branch Z metamodel. The relations in the comparison are expressed in terms of mappings. The conceptual mappings can facilitate the concrete mappings between models. For example, in Listing 4.2, class *Person* in the company Y metamodel can be mapped to class *Employee* or class *Developer* in the branch Z metamodel. On the model level, we need to find which specific person can be mapped to employee, and which should be mapped to developer.

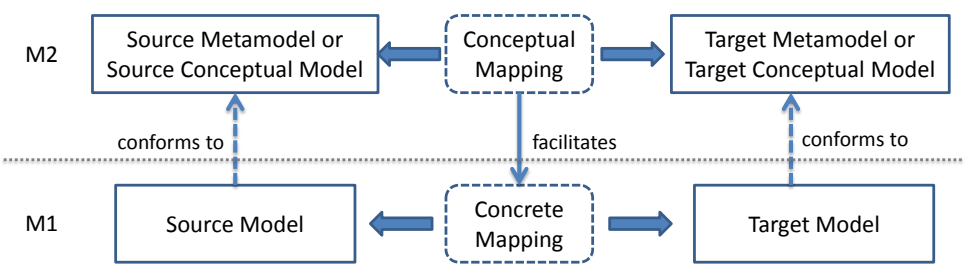


Figure 4.2: An illustration of mapping support.

Listing 4.1: The mapping information from the domain metamodel (DMM) to the company X metamodel (comXMM): The class *Person* in the DMM is renamed to the class *Employee* in the comXMM, thus the mapping type between these two classes are full mapping (line 3-7). The class *Contract* in the comXMM is a newly added class, then there is no mapping for this class (line 8-12). The class *Resume* in the comXMM is copied from the DMM, so this class is fully mapped to the class *Resume* in the DMM (line 13-17). The definition of mapping types and implementation details are given in Section 4.3.2.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <mappings>
3    <mapping mappingType="0" sourceClass="domainMetamodel.Person"
4                         targetClass="comXMetamodel.Employee">
5      <operator>Rename</operator>
6      <reason>Rename an existing class</reason>
7    </mapping>
8    <mapping mappingType="3" sourceClass="null"
9                         targetClass="comXMetamodel.Contract">
10     <operator>AddClass</operator>
11     <reason>null</reason>
12   </mapping>
13   <mapping mappingType="0" sourceClass="domainMetamodel.Resume"
14                        targetClass="comXMetamodel.Resume">
15     <operator>NotModified</operator>
16     <reason>The class is copied from original MM.</reason>
17   </mapping>
18 </mappings>
```

4.3.2 Mapping Types: between concepts and between instances

Relations between concepts indicate whether concepts in their definition or nature are the same or related. Relations between instances also include the instantiation or content of the concepts. In this chapter, for consistency, we use comparative mappings to represent relations. For example, hazard concepts from Mil Std 822E [17] and the ISO 26262 standard can relate with each other on a conceptual level. However, if the underlying severity categories are different, the relation should indicate only a partial mapping. Essentially, we identified four types of mapping between concepts:

- Full mapping: the elements in the mapping are identical. In a reuse scenario, this implies that the characteristics of the element (such as its form, its attributes, and its references) are not changed.
- Partial mapping: there are some similarities between the elements. The similarities and differences can be analyzed and documented in the mapping document.
- Possible mapping: there is a possibility that a mapping can be found between the elements. Further analysis of the elements needs to be done to make sure that there are some similarities between them.
- No mapping: there is insufficient similarity between the elements to permit a mapping to take place.

In our implementation, we use “0”, “1”, “2”, “3” to represent full mapping, partial mapping, possible mapping, and no mapping respectively. For example, in of Listing 4.1 (line 3), the mapping type “0” means that the mapping between the class *Person* in the

Listing 4.2: The result of comparison between the company Y metamodel (comYMM) and the branchZ metamodel (ZMM): One full mapping is found between the class *Resume* in the comYMM and the class *Resume* in the ZMM. One partial mapping is found: the mapping from the class *Person* in the comYMM to the class *Employee* in the ZMM. Three possible mappings are discovered: the mapping from the class *Employee* in the comYMM to the class *Developer* in the ZMM, and to the class *Employee* in the ZMM, and the mapping from the class *Person* in the comYMM to the class *Developer* in the ZMM. The definition of mapping types and implementation details are given in Section 4.3.2.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<mappings>
  <mapping mappingType="2" sourceClass="comYMetamodel.Employee"
                      targetClass="branchZMetamodel.Developer">
   <reason>Intermediate concept is:domainMetamodel.Person</reason>
  </mapping>
  <mapping mappingType="2" sourceClass="comYMetamodel.Employee"
                       targetClass="branchZMetamodel.Employee">
   <reason>Intermediate concept is:domainMetamodel.Person</reason>
  </mapping>
  <mapping mappingType="2" sourceClass="comYMetamodel.Person"
                      targetClass="branchZMetamodel.Developer">
   <reason>Intermediate concept is:domainMetamodel.Person</reason>
  </mapping>
  <mapping mappingType="1" sourceClass="comYMetamodel.Person"
                       targetClass="branchZMetamodel.Employee">
   <reason>Intermediate concept is:domainMetamodel.Person</reason>
  </mapping>
  <mapping mappingType="0" sourceClass="comYMetamodel.Resume"
                        targetClass="branchZMetamodel.Resume">
   <reason>Intermediate concept is:domainMetamodel.Resume</reason>
  </mapping>
</mappings>
```

domain metamodel and the class *Employee* in the company X metamodel is full mapping. The reason for that is the class *Employee* is renamed from the class *Person* without changes. Moreover, the mapping types also indicate priority for conflict resolution where Level 3 is the highest. If the priority level of an existing mapping between two concepts is lower than a newly discovered mapping, the priority level between those two concepts will be changed to the new one.

4.4 Results

As mentioned in Chapter 3, the generic metamodel of safety standards oversimplifies the modeling needs of safety engineers and assessors. This can lead to over-generalization, additional manual work, and less support for automatic consistency checks. To address this, extended metamodels are refined from the GMM [145]. For our demonstration, we use a family of metamodels from safety-critical domains. All of these metamodels are derived from the GMM. Figure 4.3 shows an extract of an ISO 26262 metamodel from the automotive domain (ISOMM). Figure 4.4 shows an extract of a DO 178C metamodel from the avionic domain (DOMM).

Both of those two metamodels are from the same aforementioned family. From these two figures, we can see that the traceability information of this family is kept in the

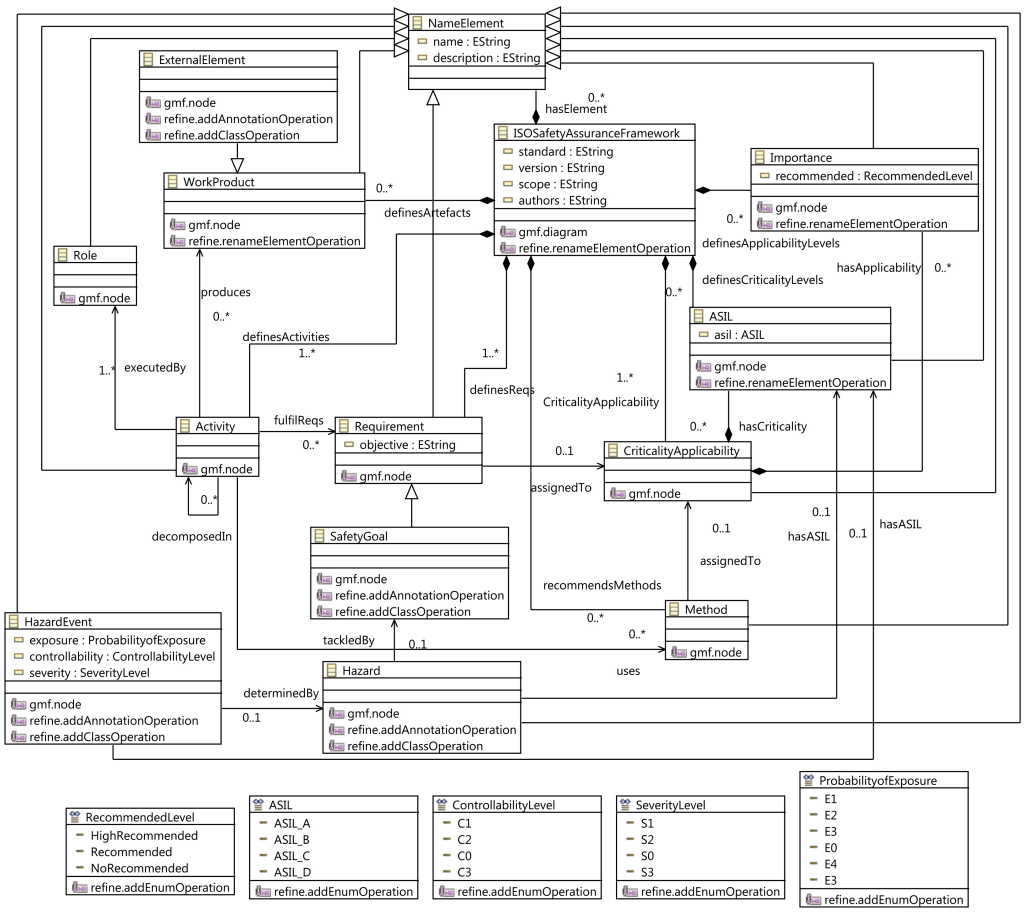


Figure 4.3: An extract of an ISO 26262 metamodel from the automotive domain.

annotations of the metamodels. The details of the annotations consist of operation name, the path of the source class, and the path of the target class. If there is a new class added, the path of the source class is assigned as the path of the source metamodel. In this way the sequence of changes are recorded in the metamodels. Only the annotations for classes are shown in Figure 4.3 and Figure 4.4. Through the metamodel transformation, this traceability information is analyzed automatically. Some results are shown as follows:

- There are five possible mappings from the ISOMM to the DOMM: from class *Importance* to class *ApplicabilityLevel*; from class *ASIL* to class *SoftwareLevel*; from class *SafetyGoal* to class *HighlevelRequirement*; from class *SafetyGoal* to class *LowlevelRequirement*; and from class *CriticalityApplicability* to class *CriticalityApplicability*.

- There are four partial mappings from the ISOMM to the DOMM: from class *ExternalElement* to class *Artefact*; from class *SafetyGoal* to class *Requirement*; from class *Requirement* to class *HighlevelRequirement*; and from class *Requirement* to class *LowlevelRequirement*.

- There are five full mappings from the ISOMM to the DOMM: from class *WorkProduct*

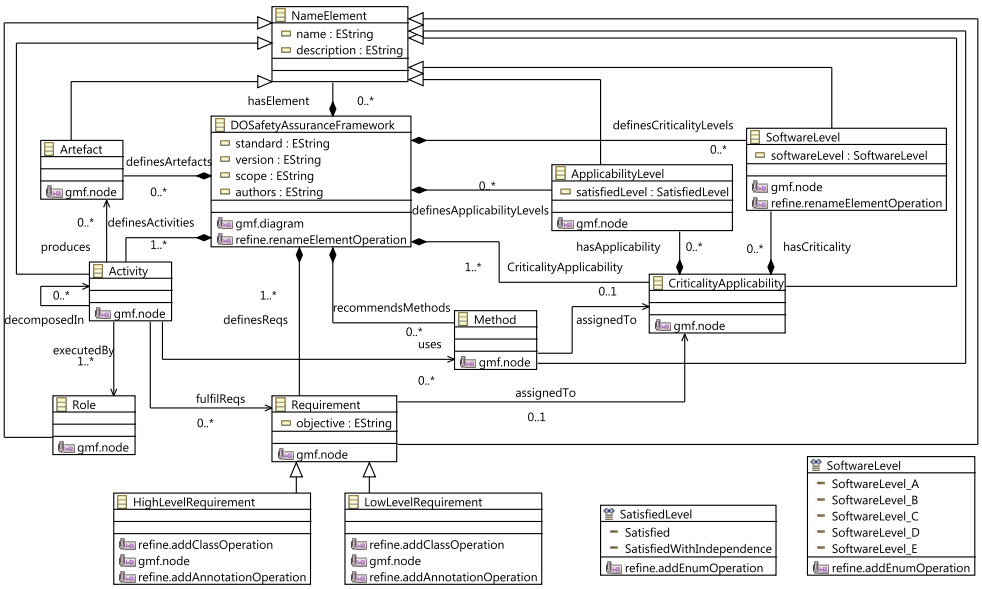


Figure 4.4: An extract of a DO 178C metamodel from the avionic domain.

to class *Artefact*; from class *Requirement* to class *Requirement*; from class *Artefact* to class *Artefact*; from class *Method* to class *Method*; from class *Role* to class *Role*.

Note that some classes are abstract, like *NameElement*. They are only used for constructing the metamodels. The comparison results of those classes are not described here. Some of the comparison results are modified after validation by domain experts. For example, there is a possible mapping from class *SafetyGoal* to class *LowlevelRequirement*. During validation, the mapping type between these two classes is changed to "no mapping".

The conceptual mappings can be used in two scenarios. Firstly, they could be used for supporting concrete mappings. A concrete mapping can be found on the model level only if there exists a conceptual mapping at the metamodel level. For instance, if there is no mapping between class *SafetyGoal* and class *LowlevelRequirement*, then at the model level, all instances of *SafetyGoal* can not be mapped to the instances of *LowlevelRequirement*. Secondly, because those metamodels are used as domain-specific conceptual models in safety cases (see Chapter 5), the conceptual mappings between them can be used for supporting safety case reuse.

4.5 Related Work

Related research is found in traceability management, metamodel matching, and model comparison.

Traceability management. A model-driven framework for traceability management, called iTrace, has been developed [135], which enables the analysis of traceability information of the different models involved in the software development cycle. Also, traceability visualization in model transformations has been done to facilitate traceability analysis [34]. In this chapter, we focus on traceability management

of metamodels rather than models, and we propose to use metamodel refinement specifications to support traceability management and the analysis of traceability information.

Metamodel matching. Metamodel matching techniques support the detection of mappings between two metamodels. Those mappings are used to generate a model transformation between two metamodels. Metamodel matching for automatic transformation generation is discussed by Falleri et al. [65]. The metamodels used for metamodel matching are created independently for the same kind of applications. However, in this chapter, the metamodels are all derived from the same original metamodel. The information of those mappings is defined in refinement specifications and stored with the metamodels in the sequence of transformations. Besides, those comparative mappings can not only be used for generating a model transformation, but also for supporting safety case reuse.

Model comparison. As mentioned before, metamodel mapping can be used to facilitate model comparison. Typically, for model comparison, to calculate the model differences, metamodel equivalences and differences have to be specified. There are a number of existing approaches for representing metamodel differences [70] [51]. A method for metamodel-assisted model comparison is proposed by Zvezdan [125]. They provided a comparison metamodel that all the models being compared must conform to. Based on this comparison metamodel, an algorithm can be performed to calculate model differences. In this chapter, our focus is only on the metamodel level.

4.6 Conclusions

As discussed in Chapter 3, a family of metamodels can be generated via metamodel transformations . The traceability information between those metamodels are documented in the MMRL specifications. As all of the metamodels are generated from one generic metamodel, those specifications can be used for comparison between those metamodels. In this chapter, we have presented our approach to take into account comparative mapping support in the context of conceptual modeling. The study of comparative mapping support between conceptual models or metamodels in safety-critical domains is a promising approach to improve the understanding between different domains or companies at the conceptual level and, consequently, the reuse of safety assurance data at the model level. In Chapter 5, conceptual models are also used for constructing safety cases. In this case, the mappings found between conceptual models can support safety case reuse.

The main contribution of this work is insight into traceability management and mapping support in the sequence of transformations. The traceability information is stored in metamodels themselves, which is easy to be maintained and analyzed. Meanwhile for each refinement of metamodels, a mapping specification is generated along with the target metamodels. A comparison between different conceptual models or metamodels can be obtained by analyzing the related mapping specifications.

Chapter 5

Using SBVR-based Controlled Language for Safety Case Construction

As safety standards are widely used in safety-critical domains, such as ISO 26262 in the automotive domain, more and more companies in these domains start to use safety cases for demonstrating the safety of their products. Therefore, it is crucial to ensure that a safety case is both correct and clear. To support this, we propose to make use of modeling techniques to facilitate safety assurance in the automotive domain. Continuing our work in Chapter 2, a rule-based approach enables us to extract a conceptual model from safety standards or project guidelines. Then, by expressing safety cases in structured English using an SBVR vocabulary, a safety case is linked to the conceptual model, and the content of it is enforced to be well structured and controlled. The contribution of the explicit link between the safety case and the conceptual model is to reduce the ambiguity of natural language and increase the confidence in the claimed safety assurance. Furthermore, an SBVR editor for building a vocabulary and a GSN editor with vocabulary support are developed. Finally, two case studies have been carried out to demonstrate our approach and show the benefits of using the controlled language for safety case construction.

5.1 Introduction

A safety case is used for assuring the desired safety and showing the confidence in the claimed safety assurance (See Section 5.2.1). Therefore, it must be trustworthy and understandable. If the context of a safety case is not clear, it may affect its correctness. Currently, safety cases are expressed in natural languages, which are unavoidably ambiguous. This sometimes undermines the confidence in safety cases [74]. Moreover, safety assessment or certification is amongst the most expensive and time-consuming tasks of safety engineering for modern systems [57]. It becomes even more challenging when a system evolves and re-assessment is needed. Therefore, more and more researchers start to work on finding more efficient and cheaper methods for safety assessment.

Modeling techniques are introduced to support safety assessment. Their contribution is

to reduce manual work by creating machine processable and reusable models in the target domain, such as the automotive or the railway domain. By decreasing the amount of manual work, modeling techniques can support reducing costs, as well as avoiding human mistakes. Furthermore, modeling techniques can be used to facilitate the development of safety argumentation and to increase the understandability and confidence in the claimed safety assurance [24].

However, there are a number of challenges: First, domain knowledge is usually implicit. When domain experts work from their own experiences, the knowledge they have is implicit. It is a challenging task to make this implicit knowledge explicit [140] [150]. Second, a number of users (argument readers and writers, such as safety engineers, or safety assessors) can contribute to construct, assess or use the safety case. The larger the number of users of a safety case, the more different interpretations of it. The use of natural language does not help to reduce the inconsistencies and ambiguities in the safety argumentation. This makes it difficult to avoid misunderstanding and mistakes, and to bridge the gaps between different users of a safety case.

Safety cases are documented in textual or graphical notations. For textual safety cases, the details of reasoning and safety argumentation can be captured in Word or Excel documents and presented in a purely linear format [6]. The logic and structure of this textual form is not easy to see, which allows for inconsistencies and confusion [151]. This motivates the development of graphical notations, such as Goal Structuring Notation (GSN) [86]. It aims at assisting developing well-structured and well-reasoned safety arguments by using basic elements, like goals, strategies, and evidence. By using GSN, the structure of a safety case is explicit and clear. However, as mentioned before, more and more users are involved in safety case development. Common understanding of the meaning of a safety case element becomes important. If it is not the case, the confidence of a safety case can be misplaced. Furthermore, safety argument patterns are introduced to effectively develop successful safety cases. Nevertheless, the statements used inside those GSN elements are still expressed in natural language.

To tackle the aforementioned challenges, some research has been done on the understandability of GSN safety arguments. Assured safety arguments are proposed as a clear argument structure to demonstrate how to create clear safety arguments [79]. Besides, a precise definition of the Context elements in GSN arguments is proposed to achieve a better understanding [74]. However, the content of a safety case element is still documented by natural language. The ambiguities caused by natural language are still unsolved. To address this, using a controlled language for constructing GSN safety case is promising.

In this chapter, we propose to a novel approach that guides system designers in establishing a sound relationship between conceptual models and safety cases. The major contribution of our approach is threefold. First, based on our rule-based approach, conceptual models of safety standards can be extracted, in which domain knowledge is well-structured (Chapter 2). Second, our approach makes use of Semantics of Business Vocabulary and Business Rules (SBVR) [116], which enables us to overcome the syntactic inconsistencies and semantic ambiguities involved in natural language expressions. The extracted conceptual models will be described in SBVR format. Later, it could be used to express the safety cases in a clear and controlled way. Besides, the understandability of safety arguments can be improved. To support the development of an SBVR vocabulary, a model transformation is proposed to generate SBVR vocabularies from EMF conceptual models. It reduces the manual work involved in vocabulary development. Moreover, we build two editors to facilitate the safety case construction. An SBVR editor is implemented

for modifying or creating a vocabulary, and a GSN editor is developed to enable argument writers to edit GSN elements in the controlled language. Finally, we carried out two case studies to demonstrate our approach and show the benefits of using the controlled language for construction safety cases.

The remaining chapter is organized as follows: Section 5.2 introduces the concept of safety case, GSN, and SBVR specification. Section 5.3 presents the three main phases in our vocabulary-based methodology for safety case development. Section 5.4 gives a brief introduction of a case study. Section 5.5 discusses the related work. Finally, Section 5.6 summarizes this chapter.

5.2 Background Information

We describe in this section the basic information used in the remainder of this chapter. First we give a short description of safety cases, and then we introduce SBVR and identify the applied terminology.

5.2.1 Safety Case and GSN

A safety case is a structured and reasoning argument, connecting claims to a body of evidence. A safety case is defined as "*a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment*" [42]. In ISO 26262, the following definition of a safety case is given: "*argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities during development.*" [12]. The safety case is used to show that a system, service or organization will operate as intended for a defined application in a defined environment. As noted earlier, a safety case must be represented both correctly and understandably. Thus, the reasoning structure of a safety case should be carefully developed and defined.

A safety case consists of three principal elements: objective (or safety claim), argument, and evidence [149]. Figure 5.1 shows the relations between these elements. A safety objective describes safety requirements for a specific system in a given context. Evidence is defined as information which indicates whether the objective has been met. An argument

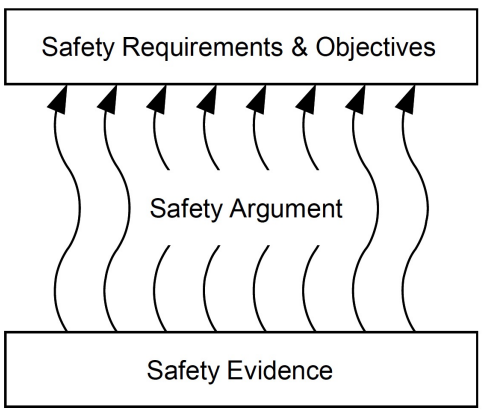


Figure 5.1: "The Role of Safety Argumentation" (Figure 1 from [86]).

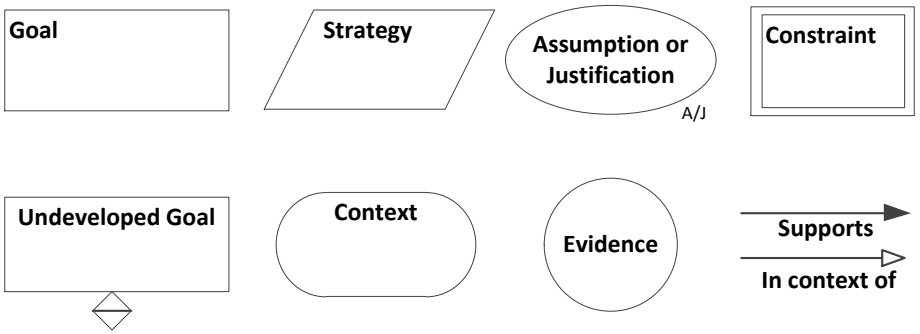


Figure 5.2: Some elements of the Goal Structuring Notation.

is a falsifiable statement of objectives based on the given evidence. To construct a safety case, top safety claims are refined into their sub-claims. Eventually, the basic safety claims can be obtained, which require safety evidence to support them. Then, safety evidence collection (see Section 6.2) is performed to collect evidence for those basic safety claims.

Based on these observations, the Goal Structuring Notation has been proposed for representing the argument structure [86]. Goal Structuring Notation (GSN) is a graphical notation which is widely recommended for presenting safety cases [88]. It provides a clear and well-structured argument in terms of basic graphical elements, such as goals, solutions and strategies. As mentioned before, safety standards such as ISO 26262 (automotive) and DO 178C (avionic), require documentation of safety cases for safety-critical systems, and GSN provides the standard format to document the safety cases graphically. The most stable and referenced documentation of GSN is called GSN Community Standard [77]. Some elements of the standard GSN are introduced in Figure 5.2.

As safety case pattern requires more flexibility and complexity on the structure, the standard GSN has been extended with several elements and entities (Figure 5.3). The detailed information of the elements can be found in the GSN Community Standard. Only the sixth element (Assurance Claim Point, ACP) is not described in the GSN Community Standards. This arrow means a confidence argument should be indicated for this connection [79].

5.2.2 Semantics of Business Vocabulary and Business Rules

Semantics of Business Vocabulary and Business Rules (SBVR) is a standard business-focused specification proposed by the Object Management Group (OMG) in 2008. It is designed for domain experts to capture business rules in a formal, structured and understandable language [137]. Recently, OMG published a second version (SBVR 1.2) [116]. The SBVR specification defines, among others, a metamodel to develop models of business vocabulary (comparable to conceptual models) and business rules (comparable to constraints that should be enforced). An extract of the SBVR metamodel is shown in Figure 5.4. The definitions of some terms used in this chapter are as follows:

Vocabulary: set of noun concepts, verb concepts, as well as various specialized concepts

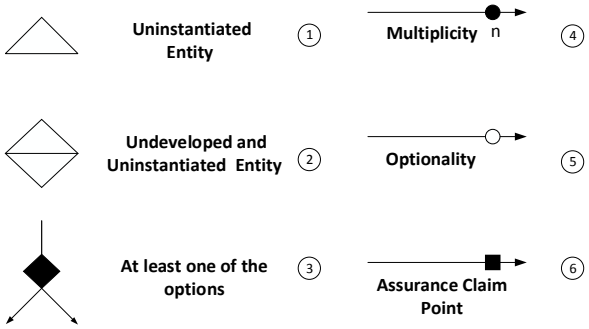


Figure 5.3: Extension of GSN to support safety case pattern.

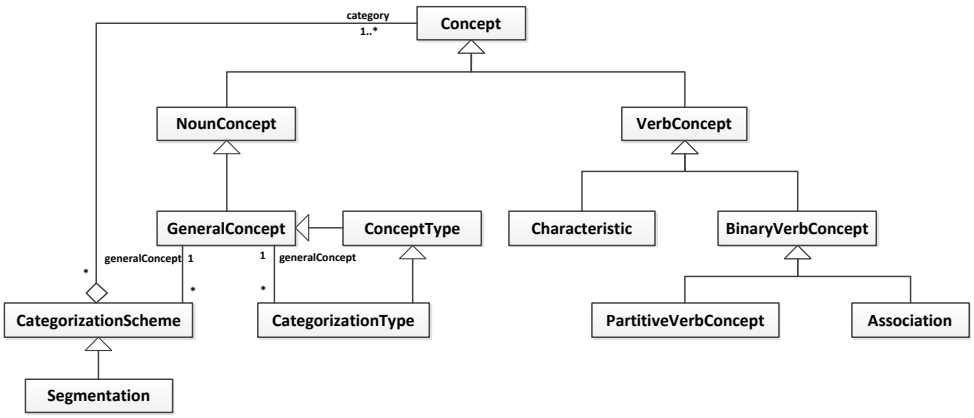


Figure 5.4: Extract of the SBVR metamodel.

such as categorizations [137].

Meaning: what is meant by a word, sign, statement, or description; what someone intends to express or what someone understands (from [116] 8.2).

Concept: unit of knowledge created by a unique combination of characteristics (from [116] 8.2.1).

Rule: proposition that is a claim of obligation or of necessity (from [116] 12.2.2).

Business rule: rule that is under business jurisdiction (from [116] 12.2.2).

In this chapter, all SBVR examples are given in SBVR Structured English (SSE), which is introduced in SBVR Annex A [116]. There are four font styles with formal meaning in SSE, which are shown in Table 5.1. In our implementation, for the font style of *Name*, we use the same font style as *Term*.

There are two key elements of SBVR meanings: business vocabulary and business rules. A business vocabulary defines concepts and their representations (designation, definition

Table 5.1: Font styles and Color with formal meaning in SBVR structured english

SSE Concepts	Font Style	Color	Denotes
Term	Underlined	Green	Noun concepts
Name	Double underlined	Green	Individual concepts
Verb	Italic	Blue	Verb concepts
Keyword	Normal	Orange	Other linguistic symbols used for definitions and statements

or statement). The concepts consist of noun concepts and verb concepts. Business rules provide elements of guidance on business structure and actions. SBVR defines deontic and alethic modalities for the formulations of guidance. The deontic modal operators describe behavioral (operative rules), which specify expectations of humans or automated systems; for example: **it is obligatory that each functional safety requirement *has* exactly one ASIL**. Alethic modal operators enable definitional structural rules, which define features of a model, thus cannot be violated, such as: **it is possible that each functional safety requirement *is derived from* at least one hazard**.

5.3 Methodology

In this chapter, we propose a vocabulary-based semi-automatic approach for constructing safety cases. An overview of our approach is shown in Figure 5.5. There are three phases: Conceptual Phase (P1), Vocabulary Phase (P2), and Modeling Phase (P3). In the conceptual phase (P1), a conceptual model of the target domain is manually built from scratch (see Chapter 2) or semi-automatically refined from other conceptual models (see Chapter 3). The conceptual model is used as an input for the vocabulary development. The metamodel that we use for describing conceptual models is an Ecore metamodel. After the creation of the conceptual model, a model transformation is applied

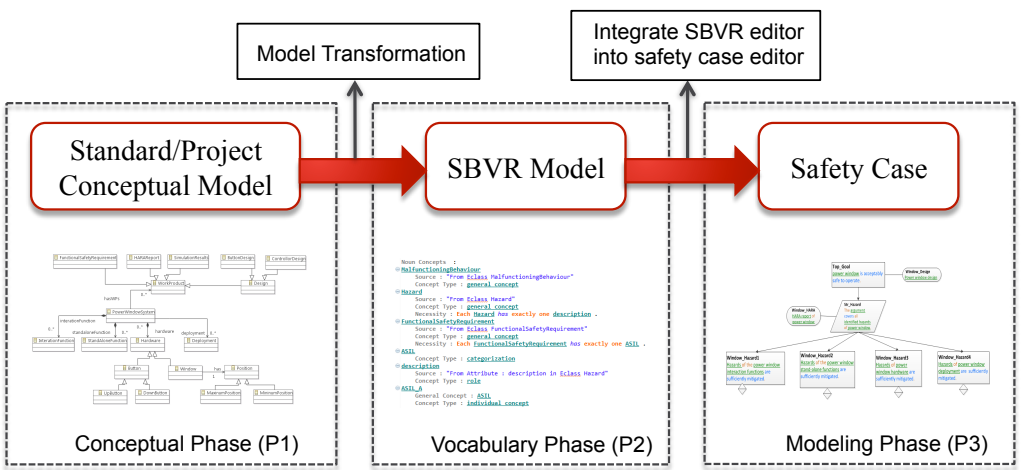


Figure 5.5: An overview of our methodology.

Table 5.2: Mapping between EMF and SBVR concepts.

EMF Concepts	SBVR concepts
Class	General concept
Enumeration Literal	Individual concept
Attribute	Role
Association	Verb concept
Generalization/Enumeration	Categorization
Multiplicity	Necessity

to transform the conceptual model from an EMF format to an SBVR specification. Then in the vocabulary phase (P2), users (argument writers) can build their own vocabulary based on the generated SBVR model. Note that, users can also skip the previous phase and start by creating a new SBVR vocabulary. Finally, in the modeling phase (P3), the vocabulary can be used to express safety cases in the controlled language. The details of these three phases are described in the following subsections.

5.3.1 Conceptual Phase: from Conceptual Model to SBVR Model

The conceptual phase is a preparation phase for a safety case. To make the domain knowledge explicit and develop a common understanding, we propose the development of a conceptual model. A conceptual model could represent the main terms and their relations that need to be considered for safety cases. Moreover, the Snowball approach in Chapter 2 can be used to facilitate modeling the standards. The approach can be used by modeling experts, but must be validated by standard or domain experts.

As mentioned before, to facilitate the formulation of safety cases, a model transformation is introduced to transform a conceptual model in the EMF format to the SBVR format. The definition of this model transformation is created based on a mapping between the concepts of those two formats (Table 5.2), and implemented in the Epsilon Transformation Language (ETL) [131]. Finally the conceptual model in SBVR can be used as an input for an SBVR vocabulary in the vocabulary phase.

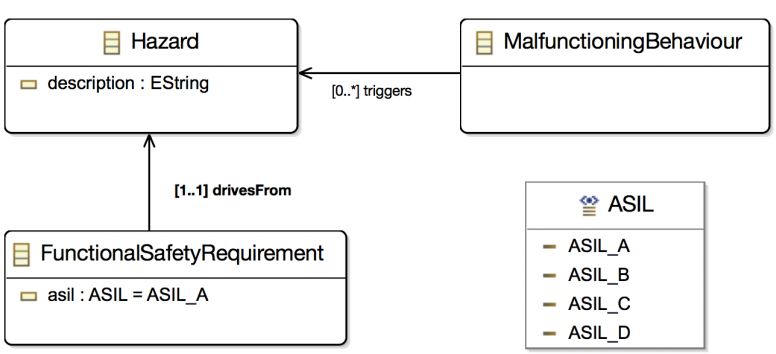


Figure 5.6: A part of ISO 26262 metamodel in EMF.

```
Noun Concepts  :
MalfunctioningBehaviour
    Source : "From Eclass MalfunctioningBehaviour"
    Concept Type : general concept
Hazard
    Source : "From Eclass Hazard"
    Concept Type : general concept
    Necessity : Each Hazard has exactly one description .
FunctionalSafetyRequirement
    Source : "From Eclass FunctionalSafetyRequirement"
    Concept Type : general concept
    Necessity : Each FunctionalSafetyRequirement has exactly one ASIL .
ASIL
    Concept Type : categorization
description
    Source : "From Attribute : description in Eclass Hazard"
    Concept Type : role
ASIL_A
    General Concept : ASIL
    Concept Type : individual concept
```

Figure 5.7: An example of SBVR Noun concepts generated from EMF concepts (screenshot of our editor).

```
Verb Concepts :
MalfunctioningBehaviour triggers Hazard
    Concept Type : association
FunctionalSafetyRequirement is_derived_from Hazard
    Concept Type : association
    Necessity : Each FunctionalSafetyRequirement is_derived_from at least one
    Hazard .
;
```

Figure 5.8: An example of SBVR Verb concepts generated from EMF concepts (screenshot of our editor).

Figure 5.6 illustrates a simple EMF schema, which is derived from an industrial ISO 26262 metamodel [21]. As the whole ISO metamodel is too big for demonstration, we select a small part from it. However, our approach can be applied to the whole ISO metamodel as well. The selected part (Figure 5.6) represents the relations between *Malfunctioning Behaviour*, *Hazard*, and *Functional Safety Requirement*. Besides it includes an enumeration type, *ASIL*. By applying the model transformation, an SBVR representation of the original conceptual model can be generated. All concepts are categorized into noun concepts and verb concepts. For the example shown in Figure 5.6, there are ten noun concepts (three instances of *General Concept*, one instance of *Categorization*, four instances of *Individual Concept*, and two instances of *Role*) and two verb concepts (instances of *Association*) derived from the EMF model. Some of the noun concepts and verb concepts are shown in Figure 5.7 and Figure 5.8 respectively. In these two figures, *Concept Type* represents the type of a noun concept or a verb concept, while *Source* shows the traceability information. The detailed information of the SBVR editor is discussed in Section 5.3.2.1.

5.3.2 Vocabulary Phase: Creating Vocabulary with an SBVR Editor

The concepts used in a safety case mainly consist of the concepts from safety standards, the concepts from a specific project, and the concepts for constructing the safety case. Therefore, to support safety case development, the generated vocabulary needs to be checked by domain experts and/or refined with additional safety-related concepts. In this phase, we developed an SBVR editor to support argument writers to build their own vocabulary. As mentioned before, the vocabulary can be completely new or a modification of an existing one. The detailed information of our SBVR editor is described in this section.

5.3.2.1 Main functions of the SBVR editor

The main functions of the SBVR editor include a vocabulary editor and a rule editor. The vocabulary editor enables users to define noun and verb concepts, while the rule editor enables users to define SBVR rules with the vocabulary. In the SBVR editor, the "**noun**" font represents general concepts, individual concepts, roles and categorizations whereas the "***verb***" font represents verb concepts. Besides, there are two types of keywords defined in our editor: default keywords and structural keywords. Both types of keywords are predefined. The default keywords are keywords defined in the SBVR specification. They are displayed in "**keyword**" font style, for example, "**each**" and "**exactly one**". The structural keywords are designed by us for the structure of a vocabulary or the characteristics of noun and verb concepts. They are shown in a gray font style. For instance, in Figure 5.7 and Figure 5.8, "**Noun Concepts**" and "**Verb Concepts**" are defined to categorize concepts into noun or verb. The field "**Concept Type**" shows the concept type of a noun or verb concept. The field "**Source**" keeps the traceability information from an EMF model to an SBVR model. Moreover, the field "**Necessity**" describes the constraints of noun or verb concepts.

There are also predefined noun concepts and verb concepts derived from the SBVR specification. The concepts in the SBVR metamodel are predefined noun concepts, such as **general concept**. The predefined verb concepts are extracted from the SBVR specification to represent some basic association types. The verb concept "***has***" or its passive form "***is_property_of***" identifies the essential properties of given noun concepts. The containment association is represented by "***includes***" (active form) or "***is_included_in***" (passive form). Besides, the categorization association is represented by "***specializes***", "***generalizes***", "***is_category_of***", or "***is_a***".

Our rule editor supports four types of model operations defined in the SBVR specification: obligation formulation, necessity formulation, possibility formulation, and permissibility formulation. A number of keywords are predefined for those model operations. "**it_is_obligatory_that**" and "**it_is_prohibited_that**" are defined for obligation formulation. "**it_is_necessary_that**" and "**it_is_impossible_that**" are defined for necessity formulation. "**it_is_possible_that**" is for possibility formulation. "**it_is_permitted_that**" is for permissibility formulation.

5.3.2.2 Graphical editor for vocabulary

Along with the textual editor, a graphical editor has been implemented to enable users to modify or build their vocabulary via a diagram. A diagram provides an overall picture of a given vocabulary, which shows the relations of the concepts clearly. As the textual and

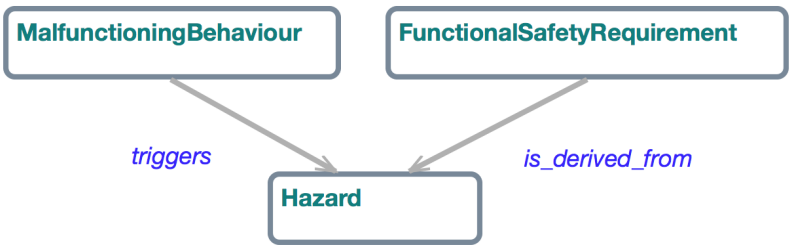


Figure 5.9: An illustration of the graphic diagram of verb concepts shown in Figure 5.8.

graphic editor are synchronized, a noun concept or verb concept can also be added or modified via the diagram. In other words, the graphical editor provides an alternative way for users to build their vocabularies.

All noun concepts and verb concepts defined in the vocabulary can be represented in a corresponding diagram. Figure 5.9 shows a diagram of verb concepts defined in Figure 5.8. We can see that, the noun concepts (**MalfunctioningBehaviour**, **Hazard**, and **FuctionalSafetyRequirement**) are shown as nodes, while the verb concepts (***triggers*** and ***is_ derived_ from***) are shown as links between those nodes. In addition, the properties of concepts can be found in the property view of the graphical element. Labels of noun and verb concepts in the diagram are displayed in the same font color as in the textual editor.

5.3.2.3 Vocabulary checking

When building a vocabulary, duplicated noun and verb concepts can be created. This causes a risk of ambiguities and inconsistencies in the vocabulary. Consequently, the understandability of the safety claims, which use those duplicated concepts, will be hampered. To address this, vocabulary checking is implemented in our editor. The goal of the vocabulary checking phase is to check the size of a vocabulary and find duplications in

Listing 5.1: An example of vocabulary checking report shows: There are 55 noun concepts defined (line 3-6), 16 verb concepts defined (line 7-10). Besides, there are two noun concpets with the same name “ASIL” (line 11-14).

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <reports>
3     <report>
4           <check> Definitions of Noun Concepts </check>
5           <reason> 55 noun concepts are defined </reason>
6     </report>
7     <report>
8           <check> Definitions of Verb Concepts </check>
9           <reason> 16 verb concepts are defined </reason>
10    </report>
11    <report>
12         <check> Duplicated Concepts </check>
13         <reason> 2 noun concepts with the same name : ASIL </reason>
14    </report>
15 </reports>
```

a vocabulary. The output of the vocabulary checking is a report, which shows the number of defined noun and verb concepts and duplicated concepts that need to be addressed. A checking report is automatically generated after a vocabulary created. An example of the checking report is shown in Listing 5.1.

5.3.3 Modeling Phase: Construct Safety Cases with Vocabulary

In the modeling phase, the safety argument will be constructed in GSN with vocabulary support. We propose to use SBVR to express the content of each safety case element. To facilitate the application of our method we have developed a tool framework. It is implemented using the Eclipse Modeling Framework with certain plug-ins: Graphical Modeling Framework (GMF), EuGENia [4], Xtext [7]. GMF and Eugenia are used for building a GSN editor for safety cases. Then Xtext is used for the editing support of SBVR in our GSN editor. By using Xtext, syntax highlighting and content assistance are also provided. The input of this tool is an SBVR vocabulary, which includes the keywords for noun concepts and verb concepts used in safety cases.

By integrating the SBVR editor into the GSN editor, the noun and verb concepts defined in a vocabulary will be highlighted when safety engineers edit a GSN element. Figure 5.10 shows a screenshot of our GSN editor. When a GSN element is edited, a list of suggested concepts is given via Content Assistant. For example, after typing "p", a list of concepts in the vocabulary that start with "p" is provided. In this way, the number of errors, such as ambiguities of a safety case can be reduced. Users can always look into the vocabulary to check the definitions of nouns and verbs used in their safety cases to avoid misunderstanding.

5.4 Case Study

In this section, we discuss two case studies carried out by two master students. One student worked on a case study on a power window system, which is used to demonstrate

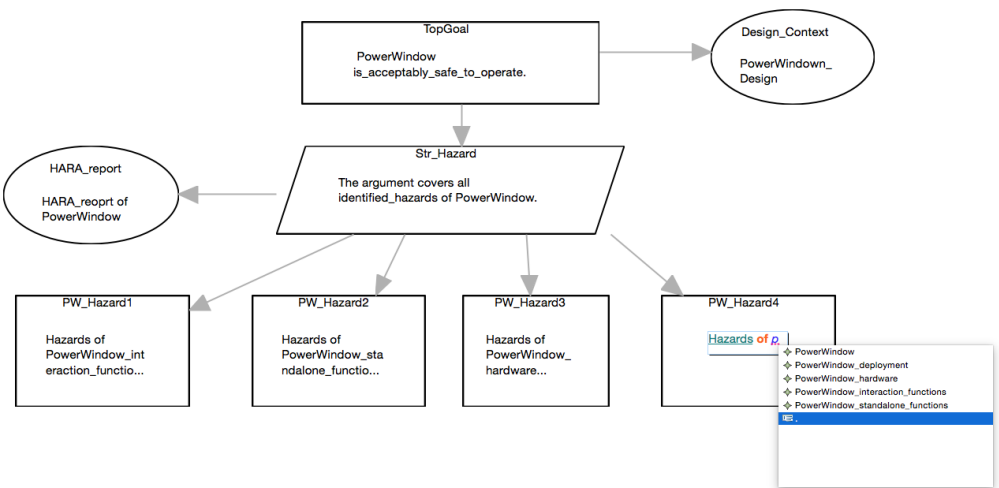


Figure 5.10: Illustration of our graphical editor.

how to apply our approach, and the other student worked on the published safety cases, which is used to find the potential benefits by using our approach. To highlight the editing GSN element and avoid disturbing users, our editor is designed to show the editing element in the SBVR format, and all other elements in plain text (See Figure 5.10). However, in this section, to demonstrate the details of the safety cases expressed in SBVR, all the elements are shown in the SBVR format [1].

5.4.1 Safety Case Construction for a Power Window System

An example of a safety case for a power window system is presented here. A power window is a window in a car that can be opened and closed by pressing a button. Power windows have the capacity to exert enough force to fracture or crush bones or even strangle a child. Therefore, when designing a power window system, its safety needs to be assured. The safety case of the power window provided for this case study is shown in Figure 5.11. The original power window safety case are created by a master student, who worked as a system engineer and designed the power window system. Then he re-built the safety case by using our approach.

Figure 5.12 illustrates the GSN and SBVR representation of the above safety case. From these two figures, we can see that some descriptions of safety case elements have been modified, for example the description of the "Window_hazard1" claim. In the original claim, "interaction with other functions" could mean interaction between functions of the power window or other parts of the power window system. Due to this ambiguity, the understandability of this claim can be affected. In Figure 5.12, it is changed to "power window interaction functions", which is defined in our SBVR vocabulary with detailed information. In this way, the ambiguity in the original sentence is reduced, and the understandability of this claim can be improved.

After comparing the whole power window safety case to the conceptual model of the power window shown in Figure 5.13, some results are obtained. First, we found that most concepts in the power window conceptual model are used in the safety case. Second, some project-related concepts in the safety case need to be added by domain experts, such as project managers. In our case, for example, "direction", "pressed button" and "limit" are introduced into our vocabulary as extra noun concepts for the power window project;

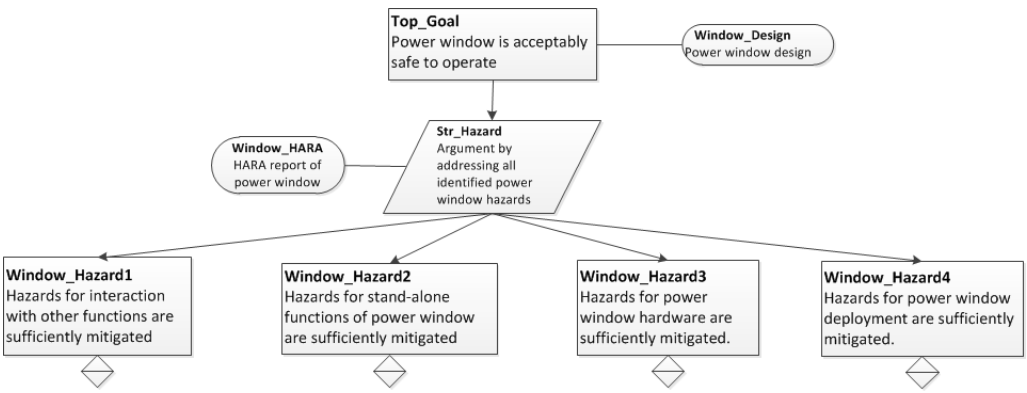


Figure 5.11: Extract of a safety case of power window.

[1] All the figures in this section are created using Microsoft Visio Studio

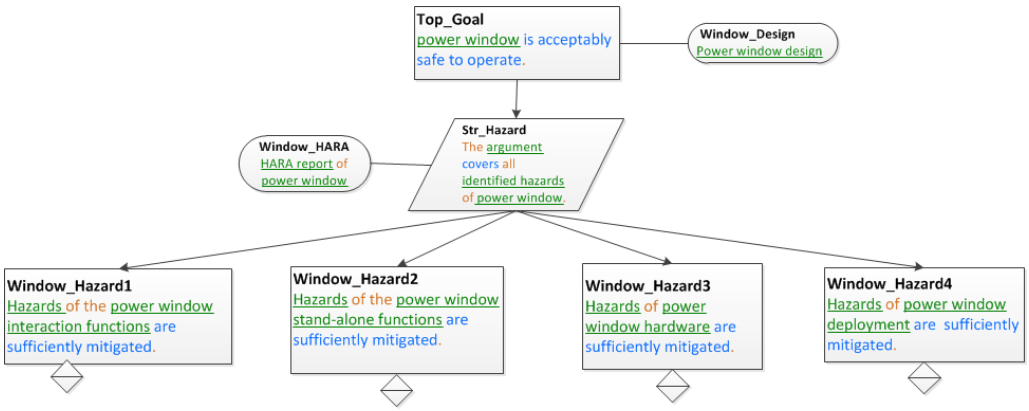


Figure 5.12: A part of power window safety case expressed in SBVR.

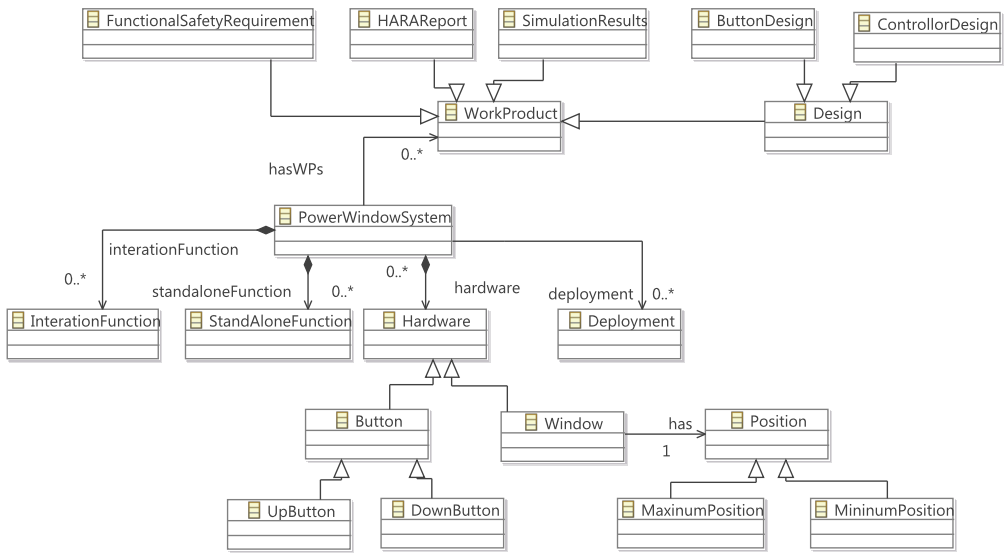


Figure 5.13: Conceptual model of power window.

"moves in", "is pressed", "moves up", "moves down", "keeps", "exceeds" are added as extra verb concepts for the power window project. Finally, some safety related concepts need to be introduced by safety case experts, such as safety assessors or safety managers. In our case, concepts such as "argument", "identified hazards", "identified functional hazards", "functional safety requirements" are added as safety related noun concepts. The concepts "is acceptably safe", "covers", and "are sufficiently mitigated" are introduced as safety related verb concepts.

Based on our metamodeling approach, the concepts used in safety cases could be categorized according to different projects or standards. Through the links between conceptual models and safety cases, information like definitions, requirements, and related concepts could be obtained.

5.4.2 Potential Benefits of Our Approach

To demonstrate how controlled language can facilitate safety case construction, we chose two published safety cases for our case study. One is a preliminary safety case for a Whole Airspace Air Traffic Management (ATM) System [9], the other is a safety case for a hypothetical Air Traffic Services Unit (ATSU) [62]. While performing these case studies, two main benefits of using controlled language for safety cases are discussed: the ambiguities in the safety case can be reduced, and the structure of the safety case can be simplified.

Reduce ambiguities in safety case. Figure 5.14 shows a part of the Whole Airspace ATM system safety case. The concepts or terms which can introduce ambiguities into the safety case are circled. We can see that different terminology is used for expressing the same thing. For example, *Area* and *geographical areas*, *ATM rules* and *Basic ATM rules*, *Assumptions for Area safety* and *safety assumptions*. For those similar concepts, some questions can be raised. Are *geographical areas* a part of or the same as the *Area*? Are *Basic ATM rules* a part of or the same as the *ATM rules*? Are *Assumptions for Area safety* a part of or the same as the *safety assumptions*?

If these ambiguities exist in a safety case, the clearness and understandability of the safety case can be affected. Consequently the confidence of the safety case may get harmed. By introducing a controlled language, this issue can be addressed. All the concepts used in the safety case have an explicit definition in the SBVR vocabulary. In this way, the ambiguities in the safety case can be reduced. Figure 5.15 shows the SBVR format of the previous safety case example (Figure 5.14).

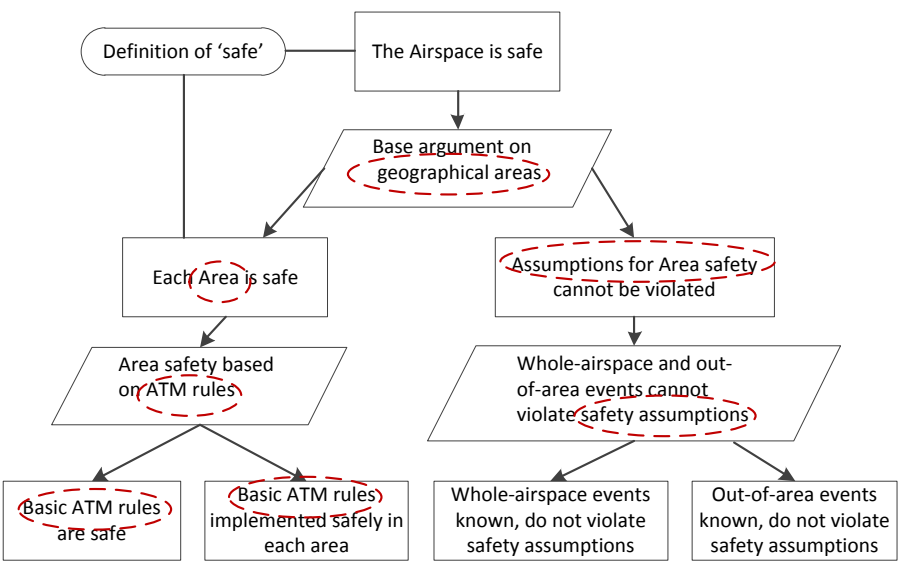


Figure 5.14: Extract from 'Complete Structure Based On Geographical Areas Strategy' (Figure 6-4 from [9]).

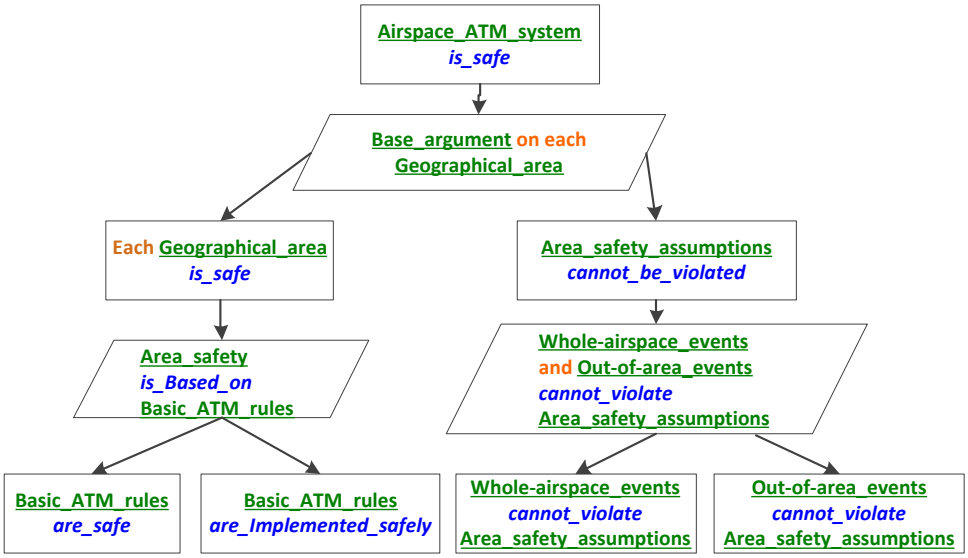


Figure 5.15: The SBVR format of the safety case example shown in Figure 5.14. The context element *Definition of 'safe'* is removed because the definition of 'safe' can be added in the SBVR vocabulary.

Simplify the structure of safety case. Sometimes, in a GSN safety case, safety engineers might add detailed information to the safety case by introducing more GSN elements, for instance a context element can be used to explain a term's meaning [77]. In Figure 5.16, a part of the overall safety argument for a Unit Safety Case is shown. We can see that there are three context elements and one constraint element added to a safety

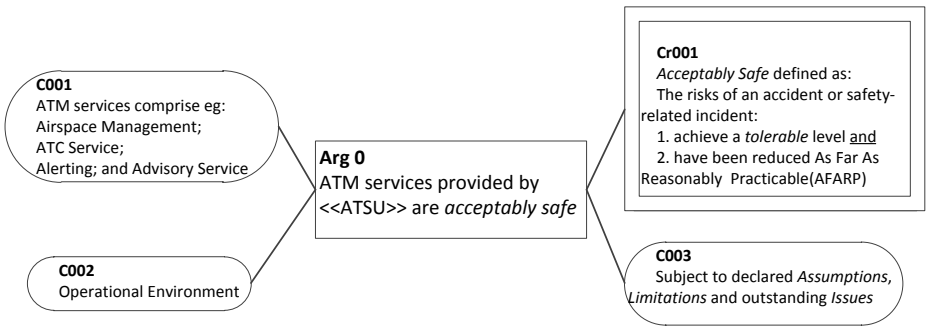


Figure 5.16: Extract from 'Overall Safety Argument for a Unit Safety Case' (Figure 12 from [62]).

Figure 5.17: The SBVR format of the safety case example shown in Figure 5.16.

claim. These context elements and constraint element can help safety case reviewers to understand the safety claim.

However, if the safety case consists of a huge number of safety claims, then elements linked to these safety claims might make the structure of the safety case more complex. Besides, some of these elements are only used to explain the meaning of the terms in a safety claim. For example, in Figure 5.16, the context element *C001* and the constraint element *Cr001* are used to provide the definition or guideline for the used terms in the safety claim *Arg0*. By using controlled language, the information or meaning of terms are provided in the SBVR vocabulary. Therefore, the safety case can be simplified. Figure 5.17 shows the SBVR format of the safety case example in Figure 5.16. We could see that by moving the definitions of the terms into the SBVR vocabulary, the structure of the safety case gets simplified.

5.5 Related Work

Tool support for SBVR. In 2006, an SBVR editor called SBeaVeR [55] was developed. SBeaVeR is an Eclipse based plug-in which implements the Structured English notation and provides a number of features: automatic syntax highlighting, automatic completion, dictionary, standard text editing. In 2010, another tool called VeTIS [112], based on the SBVR 1.0 metamodel, was created. Concerning the editing and the validation of SBVR business rules, the VeTIS tool provides the same features as SBeaVeR. Both tools provide the main feature for editing SBVR vocabulary, however the vocabulary can only be created from scratch, whereas in our approach the vocabulary can be derived from a conceptual model. Therefore, if users are not familiar with SBVR specification, they can derive an SBVR vocabulary via a conceptual model. Moreover, if users have an existing conceptual model, they could reuse it for creating the SBVR vocabulary.

Tool support for safety case construction. There exist a number of open source and Eclipse-based safety case editors, such as AdvoCATE [56], D-case [104], and CertWare [39]. Besides typical features, they also support safety case modules, patterns, and other advanced functions. In this chapter, our focus is on the controlled-language support. Our GSN editor only supports basic features for creating safety cases. However, our vocabulary editor can be integrated into other existing Eclipse-based safety case editors. In this way, our approach can be used for other applications.

SBVR and other techniques. Since the SBVR was proposed by OMG, many approaches based on it have been applied to facilitate software engineering. Most of

those approaches focus on transforming business rules or requirements written in natural language to SBVR rules through Natural Language Processing [48] [113] [38]. Some researchers worked on generating UML class models or execution models from SBVR models [33] [112]. A model transformation from SBVR to UML with OCL constraints is provided by using ATL (ATL Transformation Language) [112]. Moreover, a study of translating UML schemas to SBVR vocabularies has been carried out [44]. As all conceptual models used in this chapter are stored in EMF format, we propose a conceptual mapping between EMF and SBVR elements for getting our SBVR vocabulary.

Lewis describes the benefits of applying information modeling techniques to the development of an electronic safety case [92]. Furthermore, Lewis has also studied the relationships within the safety case. The domain knowledge is implicitly included in the informal relationships within the safety case. In this chapter, the relationships within the safety case can be found through our SBVR vocabulary.

5.6 Conclusions

In this chapter, we presented a vocabulary-based semiautomatic approach for safety case development. There are six main contributions of our approach:

1. By using SBVR vocabulary, an explicit connection between conceptual models and safety cases is created to ensure that certification data is built properly and can be reused efficiently.

2. By utilizing SBVR, the content of safety case elements is well-structured and well-controlled. It can reduce mistakes and misunderstanding between the different roles involved in producing, assessing, and using the safety case.

3. A model transformation from an EMF model to an SBVR model is proposed. A vocabulary can be automatically generated from a conceptual model, which facilities the vocabulary creation.

4. An SBVR editor is implemented. This editor supports the development of SBVR vocabularies and rules, and provides graphical editing and checking of a vocabulary.

5. A GSN editor with SBVR vocabulary support is implemented . This editor enables users to construct safety cases with their own SBVR vocabulary.

6. A case study has been carried out to demonstrate our approach and show the benefits of using controlled language for safety case construction. The results show that by using controlled language, the ambiguities in the safety case can be reduced and the structure of the safety case can be simplified.

Thus, our method supports improving the clarity and correctness of safety cases, and increasing the confidence in the claimed safety assurance. Moreover, by using our approach, the structure of safety cases can be simplified.

Chapter 6

Safety Evidence Collection

Typically the development process of safety-critical systems needs to be compliant with a number of safety objectives. To show that the required safety objectives are met, it is necessary to collect safety evidence in the form of consistent and complete data. However, manual safety evidence collection is usually tedious and time-consuming, due to a large number of artifacts and implicit relations between them. The potential ambiguities in the textual description of safety objectives even increase the difficulties of collecting the necessary safety evidence. Consequently, suppliers, who have to ensure that the required objectives have been fulfilled, need to investigate safety evidence requirements very carefully and rigorously to avoid collecting any ineffective information, or missing any important information. This chapter proposes a systematic, model-based approach to facilitate manual safety evidence collection with clear evidence requirements. To evaluate the effectiveness of our approach, an industrial case study on an avionics Real-Time Operating System (RTOS) is conducted. A large number of evidence items are collected from thousands of artifacts (involving more than 10,000 test cases and nearly thousand pages of requirement specification), for demonstrating the compliance of system development with the avionic safety standard RTCA DO-178C.

6.1 Introduction

In the safety-critical domain, the main task for safety certification of a system is demonstrating the compliance with safety standards (see Chapter 2). In the avionic domain, a series of standards, such as RTCA DO-178C [14], have been established to ensure the safe operations of systems. The demonstration of compliance with a standard requires collecting convincing safety evidence that indicates the safety objectives (claims) have been met [121]. In a real case, safety evidence information is collected from thousands of documents, source files, and test logs by suppliers and provided to safety assessors. However, manual safety evidence collection is usually time-consuming, tedious and error-prone. Without a clear understanding of safety objectives, invalid information could be collected as safety evidence. Consequently, the confidence in the evidence might decrease

and the correctness of safety assessment can not be guaranteed.

Safety evidence collection has become a crucial factor for improving the confidence on the evidence, supporting safety argumentation, and reducing the certification cost. Motivated by the challenges above, methodologies can be found on safety evidence collection and management [120], [121]. Rajwinder [121] proposed a model-driven approach based on UML profiles to present evidence requirements in safety standards. Those UML profiles are manually constructed by reviewing the IEC61508 standard [11], and they do not address the evidence collection.

In this chapter, we propose a systematic, model-based approach for collecting safety evidence. Based on the Chapter 5, a safety case is expressed in SBVR-based structured English [116]. We begin from receiving a set of basic safety objectives from the safety case. Then, Safety Objective Conceptual Models (SOCMs) are extracted from these objectives according to a number of predefined rules. Meanwhile, traceability between the extracted elements in SOCMs and the original safety claims is established. Based on this, evidence items potentially shared by safety claims can be found. After that, all SOCMs are combined into one unified model—Safety Case Conceptual Model (SCCM), in which the redundant and inconsistent concepts in those SOCMs are removed. Finally, the SCCM is used as a guide to help suppliers to collect evidence items from the artifacts produced in the development process.

For demonstration of our approach, an industrial case study on assessing the safety of Aircraft Real-Time Operating System (RTOS) is introduced. The goal of this case study is to evaluate whether the RTOS development process is compliant with the DO-178C [14]. As a result, over 20 different types of evidences and the respective relations between them are extracted from thousands of documents and database forms. Moreover, about 200 missing evidence artifacts were detected and fixed in an early phase of the certification process.

The remainder of this chapter is organized as follows: Section 6.2 introduces the background knowledge on safety evidence collection. Section 6.3 presents the details of our systematic approach for collecting evidence. Section 6.4 applies the approach to an industrial case study. The related work is covered in Section 6.5. Finally, Section 6.6 concludes this chapter.

6.2 Safety Evidence Collection

Safety evidence is defined as artifacts or information that support the argument of specific safety objectives (claims) [110]. The definition reveals two main characteristics of safety evidence: evidence is extracted from artifacts produced in a project, and used to support a safety argument. In general, a large amount of artifacts can be produced in a project, and not all the artifacts can provide information to support the argument. The process of identification and extraction of safety evidences from existing artifacts is called safety evidence collection.

Normally, safety evidence collection consists of four main phases: requirement analysis, evidence collection preparation, evidence collection, and evidence validation. In the phase of requirement analysis, the primary goal is to analyze safety objectives and identify required evidences. The output are evidence requirements. During the evidence preparation, the evidence requirements are refined to identify the relevant artifacts of a project as the data source. The output is an evidence collection guide. During the collection of evidence, data items that provide information to construct evidence

items in the identified artifacts are extracted. Finally, evidence is validated to ensure the completeness, correctness, and consistency of the collected evidences. As evidence validation has to be done manually by domain experts, it is not discussed in this chapter.

However, safety evidence collection is usually a challenging task [63]. Safety objectives in textual standards are amenable to subjective interpretations. Evidence requirements are usually presented in textual documents, and thus potentially ambiguous and inconsistent. Moreover, evidence needs to be firstly identified from a huge amount of artifacts produced during the development of a system, especially for a large-scale system developed in industry. Finally, potential problems in the original data reduce the effectiveness of collected safety evidence. This chapter addresses those issues by presenting a model-based approach for collecting safety evidence.

6.3 Methodology

As mentioned before, a systematic approach is proposed to support evidence collection from large amounts of artifacts. An overview of our approach is shown in Figure 6.1, presented as a flow diagram.

There are four phases in Figure 6.1: evidence requirement analysis (Section 6.3.1), traceability management (Section 6.3.2), evidence collection preparation (Section 6.3.3), and evidence collection (Section 6.3.4). Note that, we introduce traceability management as a new parallel phase in the safety evidence collection process. Besides, after these phases, the validation of the collected evidence needs to be carried out by domain experts. In the evidence requirement analysis phase, SOCMs are extracted from safety claims to identify the required evidence types and relations among them. Meanwhile, in the traceability management phase, the traceability between evidence requirements and the original safety claims is maintained by means of a generated traceability table. After this, in the phase of evidence collection preparation, all SOCMs are combined into an SCCM, which is used as a guide for evidence collection. Finally, evidence items for a specific

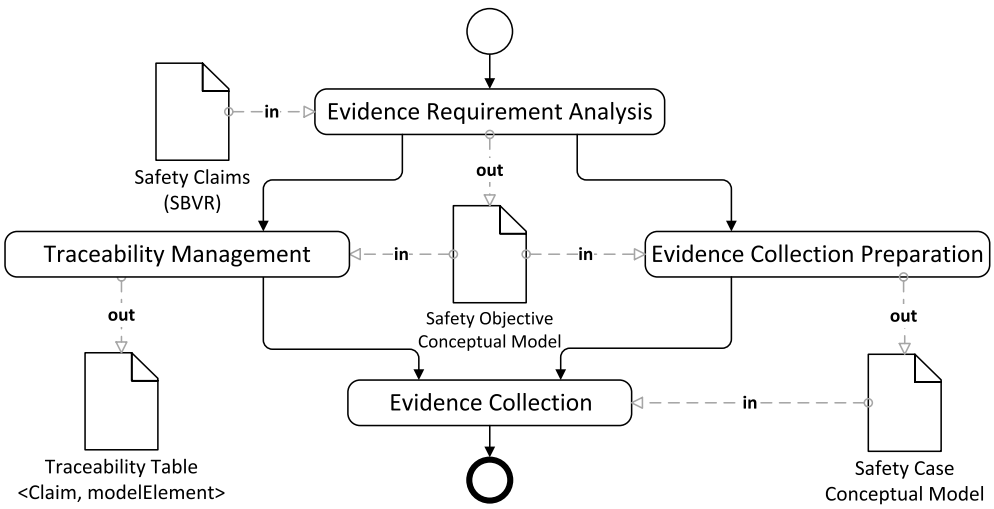


Figure 6.1: A BPMN diagram of our systematic and model-based approach for safety evidence collection.

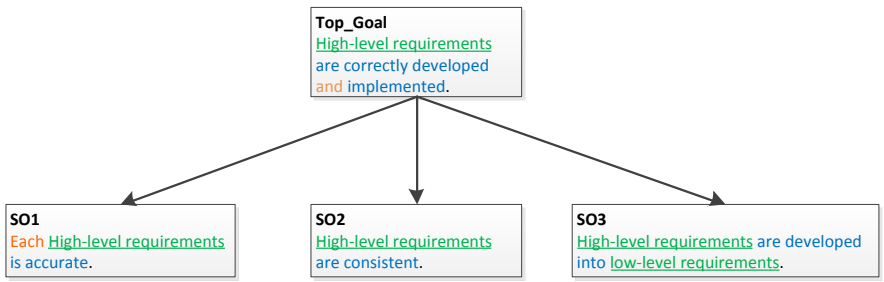


Figure 6.2: Example of objectives in DO-178C.

system are collected according to the SCCM.

To illustrate the proposed approach, we select three basic safety objectives (safety claims) in a safety case (shown in Figure 6.2), developed according to the *Software Verification Process section* of the DO 178C standard [14]. The top goal of this safety case is to argue whether high-level requirements have been correctly developed and implemented. All of these safety objectives at the bottom of the safety case require evidence to support.

- **SO1**: Each high-level requirement *is accurate*.
- **SO2**: High-level requirements *are consistent*.
- **SO3**: High-level requirements *are developed into* low-level requirements.

6.3.1 Evidence Requirement Analysis: Extracting Safety Objective Conceptual Models from Safety Claims

In this phase, evidence requirements (or SOCMs) are extracted independently from each safety objective, and presented as UML class diagrams. To facilitate the automation of our approach in the future, we only use a subset of the UML metamodel, which includes classes and associations. Consequently, two groups of rules are defined and refined though the case study to automatically extract a UML diagram from objectives in SBVR Structured English: one for obtaining classes and their attributes, the other for obtaining associations. As the work in this chapter is the results of the collaboration between two teams, the rules are firstly defined by our team (TU/e) based on existing case studies, then the team from Beihang University helped us on the refinement of the rules according to the case study shown in Section 7.4. The rules for obtaining classes and their attributes include:

- **CR1 (Noun Concept to Class)**: A noun concept in SBVR is mapped to a class in the UML diagram. For example, the noun concept "high-level requirement" can be mapped to the class *HighLevelRequirement*. We do not cover the general concepts in this rule, as the general concepts are not specific for evidence collection, For example, "high-level requirement" has a general concept "requirement", but the "requirement" will not be mapped to a UML class.

- **CR2 (Individual Concept/Role to Class)**: Individual noun concepts represent proper nouns, such as "**Hazard X**". The individual noun concept is mapped to Object. Meanwhile, the general type of the individual noun concept is added as Class. For example, the general type of "**Hazard X**" is "**Hazard**", and class *Hazard* is added into the output UML model.

- **CR3 (Characteristic to Boolean Attribute)**: Each unary verb (or characteristic) is mapped to a Boolean attribute of a Class. The Boolean value of an attribute indicates whether the subject has executed the verb or not. For example, in "**Hazards *are mitigated***", Boolean attribute "*areMitigated*" is generated for unary verb "***are mitigated***" in class *Hazard*.

The rules for obtaining associations are as follows:

- **AR1 (Keyword between Nouns to Association)**: The keywords between noun concepts are mapped to binary associations between classes. For example, in "**hazards for hardware**", the keyword "**for**" is translated to association between class *Hazard* and *Hardware*.

- **AR2 (Verb Concept to Association)**: Each binary verb concept is mapped to an association between classes. For example, in "**Low-level requirements *are traceable to* source code**", the verb concept "***are traceable to***" is mapped to the association between class *LowlevelRequirement* and *SourceCode*. The direction of generated association is from subject class to object class.

- **AR3 (Quantification to Multiplicity)**: In SBVR Structured English [116], several kinds of quantifications are defined: at least n, at most m, exactly n and universal. At **least n** quantifier is mapped to "n...*"; at **most m** quantifier is mapped to "0...m"; **exactly n** quantifier is mapped to "n"; **universal quantifiers**, such as "**each**" could be seen as exactly n quantifier, in which n equals the total number of all objects. The multiplicity would be generated when the model involves associations.

- **AR4 (Characteristic to Self-association)**: Unary verbs (or characteristics), which present the property indicating association among the instances of subject, can be mapped to a self-association class of an existing class according to the context. For example, characteristic "***are consistent***" in the context of "**High-level requirements *are consistent***" can be mapped to a self-association class *ConsistencyChecking* of the class *HighLevelRequirement*.

The output SOCMs of our example, obtained by applying the rules above, are shown in Figure 6.3 and Figure 6.4. According to the rule **CR1**, the classes *HighLevelRequirement* and *LowLevelRequirement*, are derived from general nouns "**high-level requirement**" and "**low-level requirement**" respectively. For safety objective SO1, the unary verb "***is accurate***" is mapped to Boolean attribute "*isAccurate*" in class *HighLevelRequirement* according to the rule **CR3**. For SO2, the unary verb "***are consistent***" is mapped into the self-reference class *ConsistencyChecking* of class *HighLevelRequirement* according to the rule **AR4**. Finally, for SO3, the binary verb "***are developed into***" is mapped to an association "*areDevelopedInto*" between class *HighLevelRequirement* and *LowLevelRequirement* using the rule **AR2**. The quantifier "**each**" is not mapped to multiplicity in this example since no associations are involved in the model of SO1.

Figure 6.3: The safety objective conceptual models for SO1 (left) and SO2 (right).

Figure 6.4: The safety objective conceptual model for SO3.

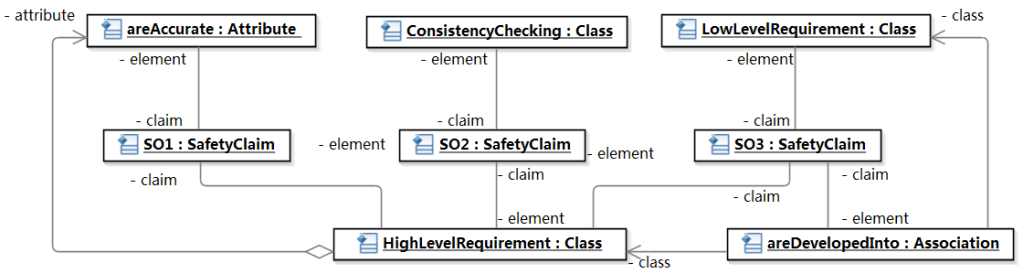


Figure 6.5: Traceability information between model elements in SOCMs and safety claims.

6.3.2 Traceability Management: From Safety Objectives to Safety Objective Conceptual Models

As mentioned before, in this phase, the traceability between the extracted concepts in the SOCMs and the original safety objectives are maintained in the form of a traceability table. Figure 6.5 shows the traceability information of our example in the UML object diagram. Class *HighLevelRequirement* and its attribute "*isAccurate*" can be traced back to SO1. Class *HighLevelRequirement* and self-association class *ConsistencyChecking* can be traced back to SO2. Moreover, class *HighLevelRequirement*, *LowLevelRequirement*, and association "*areDevelopedInto*" can be traced back to SO3.

One benefit of traceability management is that the shared evidence items can be found by searching the common model elements shared by different safety claims. For example, class *HighLevelRequirement* is traceable to three objectives. In other words, high-level requirements are potential shared evidence items among these claims.

6.3.3 Evidence Collection Preparation: Combining All Safety Objective Conceptual Models

As each SOCM is derived from each safety objective independently, in this phase, all SOCMs are combined into one SCCM to remove redundant information. For example, class HighLevelRequirement is repeatedly presented in the SOCMs of SO1-SO3. Some rules for model combination are defined as follows:

- **R1**: Classes with the same name are converged. The attributes and associations owned by the classes, that are identified as the same, are handled by using R2 and R3.

- **R2**: The attributes of a class with the same name, are combined to one. However, if types of these attributes are different, a conflict is reported.
- **R3**: The associations between the same pair of classes are merged to one, if the names of the associations are same. If multiplicities of the associations conflict with each other, errors are reported.

After applying these rules to the SOCMs for SO1-SO3, a SCCM is generated (Figure 6.6). The classes *HighLevelRequirement* in the SOCMs of SO1 and SO2 are converged to one class according to the rule **R1**.

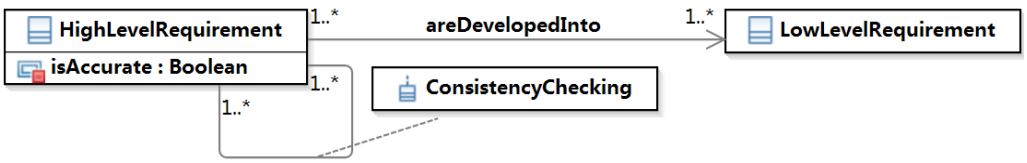


Figure 6.6: The SCCM for the running example.

6.3.4 Evidence Collection: Using the Safety Case Conceptual Model as a Guide

Artifacts that provide evidence information need to be identified and collected to support safety argumentation for a specific project. The SCCM from the previous phase, presenting the evidence requirements of the safety case, is used as a guideline for evidence collection to identify these artifacts. Thus, there are three types of evidence collection activities defined according to the SCCM.

- **Class-based Evidence Collection:** The classes in the SCCM are used to search for the relevant artifacts produced in a project lifecycle. For example, class *HighLevelRequirement* implies that the high-level requirement specifications should be collected.
- **Attribute-based Evidence Collection:** The attributes in the SCCM are used to search for information that describes the required status or characteristics of the collected artifacts. For example, attribute "*isAccurate*" in class *HighLevelRequirement* suggests suppliers to collect evidence which indicates whether a high-level requirement is accurate, such as requirement review reports.
- **Association-based Evidence Collection**: The associations in the SCCM are used to find potential relations between these collected artifacts. For example, the association "*areDevelopedInto*" represents the relation between high-level requirement specifications and low-level requirement specifications.

Table 6.1 represents the guideline of evidence collection for our example. The first two columns present type and name of model elements in our SCCM, and the third column presents suggestions on which artifacts need to be collected as evidence. Besides, in this phase, human interpretation of the guideline is needed.

Table 6.1: Evidence Collection under Guidance of SCCM.

Type	Model Element	Sample Artifacts to be Collected
Class	HighLevelRequirement	Software High-Level Requirement Specification
Class	LowLevelRequirement	Software Low-Level Requirement Specification
Attribute	isAccurate	High-Level Requirement Review Report
Association	ConsistencyChecking	High-Level Requirements Review Report
Association	areDevelopedInto	Trace Map between High-Level Requirement Specification and Low-Level Requirements Specification

6.4 Industrial Case Study: Evidence Collection for RTOS Compliance with RTCA DO-178C

Real-Time Operating Systems (RTOS) have become an essential part of avionic systems to provide environments for executing application-specific tasks concurrently and predictably [152]. In this section, we apply our approach to facilitate the certification of an avionic Real-Time Operating System, which is developed by a supplier in the avionic domain. Our goal is to argue that the RTOS development process is compliant with the avionic standard—RTCA DO-178C [14]. For the case study, we developed a safety case to argue that the software complies with safety requirements (including high-level and low-level requirements). Finally, a number of basic safety objectives of the safety case need to be supported by evidence. We choose seven safety objectives from these safety objectives for demonstrating the effectiveness of the proposed approach in the case study.

SC1: System requirements *are developed into* high-level requirements.

SC2: High-level requirements *are developed into* low-level requirements.

SC3: Low-level requirements *are developed into* source code.

SC4: Test results of normal test cases for high-level requirements *are passed*.

SC5: Test results of normal test cases for low-level requirements *are passed*.

SC6: Test results of robust test cases for high-level requirements *are passed*.

SC7: Test results of robust test cases for low-level requirements *are passed*.

The artifacts involved in our case study are produced in the development process and the testing process. In the development process, requirement specifications (including system/ high-level/ low-level requirements), design modules in software architects and functions in source files (.c) are produced. In the testing process, test programs (.c file), and test log files (.log) generated after the execution results are involved. Three kinds of

test programs are designed: functional test for verifying the compliance of functions with low-level requirements, configuration test for verifying the correct interactions between modules, and system test for verifying the correct operations provided by software with high-level requirements.

6.4.1 Model Extraction and Model Combination

By applying our approach to the case study, seven SOCMs are extracted from these selected safety claims. Then, all SOCMs are combined into one SCCM according to the combination rules, which is shown in Figure 6.7.

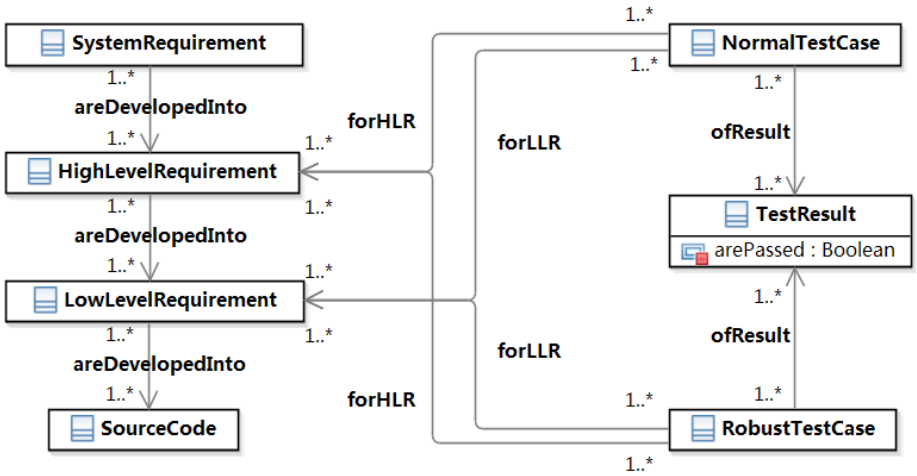


Figure 6.7: The safety case conceptual models in our case study.

In Figure 6.7, class *SystemRequirement*, *HighLevelRequirement*, *LowLevelRequirement*, and *SourceCode* are generated from noun concepts in the leaf objectives SC1-SC3. Associations "areDevelopedInto" between these four classes are derived from the binary verb concept "***are developed into***". Class *NormalCase*, *RobustTestCase*, *TestResult* are generated from the noun concepts in objectives SC4-SC7. The attribute "arePassed" in class *TestResult* is generated from the characteristic "***are passed***".

6.4.2 Traceability Management

As mentioned in Section 6.3, traceability information between safety claims and evidence requirements is maintained while combining models. Figure 6.8 shows the traceability information for safety claims SC1-SC3. Safety claim SC1 is traceable to class *HighLevelRequirement*, *SystemRequirement* and the association between them. The class *LowLevelRequirement* is traced back to both SC2 and SC3, which indicates that the low-level requirement specifications are shared evidence items between SC2 and SC3.

6.4.3 Collecting Evidence

In this phase, the SCCM is used to facilitate the evidence collection for the safety case of RTOS. The results from the class-based evidence collection for the real-time operating system are summarized in Table 6.2, where the column Class lists the classes in SCCMs,

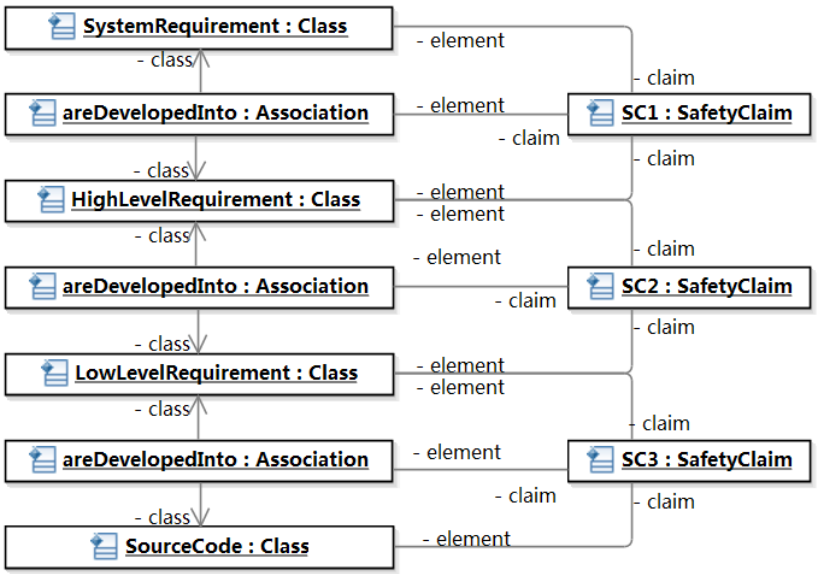


Figure 6.8: Traceability information for objective SC1-SC3.

Table 6.2: Class-based evidence collection results.

Class	**Artifacts to be collected**	**# Inst**
SystemRequirement	System Requirement Specifications (.doc)	65
HighLevelRequirement	High-level Requirement Specifications (.doc)	705
LowLevelRequirement	Low-level Requirement Specifications (.doc)	3201
SourceCode	Functions (.c)	2076
NormalTestCase	Normal Configuration Test Cases (.c) Functional Test Case (.c)	5186 12408
RobustTestCase	Robust Configuration Test Cases (.c)	2030
TestResult	Test Log Files (.log)	497
TestProcedure	Functional Test Programs (.c) Configuration Test Programs(.c)	1790 642

the column Required Artifact lists the artifacts which need to be collected, and #Inst means the number of instances found.

The relations among the collected evidence items can be derived by means of association-based evidence collection. The results of this type of collection are present in Table 6.3. For example, the association "*areDevelopedInto*" between *HighLevelRequirement* and *LowLevelRequirement* indicates the evidence of the traceability information between high-level requirements and their derived low-level requirements need to be collected.

Another benefit of our approach is the ability to systematically help suppliers to detect missing evidence items, based on the generated SCCMs. For example, there is no relevant artifact found for the association between RobustTestCase and HighLevelRequirement in Table 6.3, which indicates there is a missing evidence artifact in the provided artifacts. Due to the neglect of suppliers, some evidence artifacts are not derived or produced, such as the association between high-level requirements and robust test cases. As a result,

Table 6.3: Association-based evidence collection results.

Source	**Target**	**Artifacts to be collected**
System Requirement	HighLevel Requirement	Trace Map between System Requirements and High-Level Requirements (.xlsx)
HighLevel Requirement	LowLevel Requirement	Trace Map between High-Level and Low-Level Requirements (.xlsx)
LowLevel Requirement	SourceCode	Trace Map between Modules and Functions (.xlsx) Trace Map between Modules and LLRs (.xlsx)
NormalTest Case	HighLevel Requirement	Trace Map between System Test Cases and High-Level Requirements (.xlsx)
NormalTest Case	LowLevel Requirement	Trace Map between Configuration Test Cases and Low-Level Requirements (.xlsx)
RobustTest Case	HighLevel Requirement	No artifacts found.
RobustTest Case	LowLevel Requirement	Trace Map between Configuration Test Cases and Low-Level Requirements (.xlsx)

the collected evidence items are insufficient. This issue could be detected through the association-based evidence collection.

Additionally, after applying our approach to the whole safety case of the Real-Time Operating System, we found 36 test logs missing in the functional test, 15 test logs missing in the configuration test, 108 test programs missing in the configuration test, and 45 coverage reports lost in the functional test. By addressing those issues, the confidence of the collected evidence can be improved. Therefore, our approach can not only facilitate evidence collection, but also support to improve the evidence confidence by finding missing evidence artifacts in an early phase. The reason for the issues found during evidence collection are listed as follows:

- **Missing Test Logs:** The problem is that some test cases could not trace back to test log files after their executions. When extracting the relations between test results and test cases, this issue is found. By analyzing the reasons, we found some test programs terminated due to the crash of software, which leads to the failure of generating test log files afterwards. This problem could cause potential risk of finding software bugs since the test logs are designed to record the failures.

- **Missing Coverage Analysis Reports:** The problem is that some functions could not be traced to coverage analysis reports. This problem was found during the process of extracting and verifying the associations between low-level requirements and coverage analysis reports. The missing of coverage analysis reports were caused by software crash during testing.

- **Missing Test Programs:** The problem is that a number of test logs could not be traced to test cases. Due to the faults in the management process, some test programs were not provided to the certifiers. As a result, this problem could lead to insufficient evidence derived.

- **Wrong Data:** The problem is that some logs of unit test cases recorded wrong function names. The reason is that some functions were altered in testing to improve

test coverage, and the code was tested in different host with different dictionaries. The problem is found during the process of extracting the associations between unit test cases and the low-level requirements.

6.5 Related Work

Other work has advocated model based approaches to facilitate evidence collection. Structured Assurance Case Model (SACM) is proposed by Object Management Group (OMG) in 2011 [25]. It provides a common framework for presenting evidence and its traceability to claims in certification. Based on the SACM, Safety Evidence Traceability Information Model (SafeTIM) [110] is proposed as a smaller and more flexible approach to present the traceability between artefact and safety objectives. However, with a specific focus on constructing evidence, the process of evidence collection is implicit.

Besides, a number of projects have addressed tool support for evidence collection [61] [5]. Those tools enable user to collect evidence from external tools for safety argumentation. But our goal is to use safety argumentation as a guideline for evidence collection. The research work that carried out by Falessi et al. contributes to evidence collection indirectly [64] [63]. A model-based approach of safety evidence planning with tool support is discussed. By using a questionnaire-based agreement process, the necessary evidence for compliance can be identified. This appraoch can be used for evidence collection, but it only focuses on the planning phase.

Some other related work can be found on safety evidence management. Typically, during the safety evidence collection, a large amount of evidence can be gathered. Most of the evidence are paper-based documents. Therefore, how to manage and structure the evidence is also a challenge. Some researchers carried out their study on managing evidence electronically [122] [52]. It could facilitate the management and reuse of the collected evidence.

6.6 Conclusions

In this chapter, we presented a model-based approach to support evidence collection in safety certification, which is a basis for semi-automatic or automatic evidence collection in the future. There are a number of benefits of our systematic approach. Firstly, the safety case conceptual model provides a model-based representation of evidence requirements, which reduces the potential ambiguity and inconsistency in textual safety claims. As a result, the risk of recollecting evidence, due to misunderstanding of evidence requirements, is mitigated. Besides, in phase of traceability management, the traceability between the safety claims and the collected evidence, can help suppliers to find the potential shared evidence items between objectives. Thus, the reusability of evidence can be improved and the cost of repeatedly collecting evidence can be reduced. Finally, the traceability information could also be used to link the collected evidence items to the corresponding objectives. Another contribution of our approach is detection of evidence artifacts missing in an early phase so that the confidence of the collected evidence can be improved.

Threats to Validity There are some threats to validity related to our approach present in this chapter. Firstly, our rules are refined and applied in the context of limited case studies. Therefore, for different case studies, more specific rules might be added. We addressed this by defining the rules in a generic level, therefore users can

map their specific rules to our rules. Secondly, the generalization in the class model is not considered. Normally, for a noun concept, it could have a general concept. Then there is a generalization relation between the noun concept and its general concept. In this chapter, we only consider the basic objectives in a safety case. We think general concepts do not contribute to evidence collection. Finally, human interpretations exist during evidence collection. Currently, our approach needs to be performed manually. For applying the rules, human interpretations and experience are still needed. However, our approach provides a guideline for non-expert users to collect evidence. Moreover, as safety claims in the safety case are written by human, the concepts used in these safety claims might be incorrect or not concrete. Therefore, the interpretation between the concepts used in the safety claims and the evidence requirements for evidence collection might affect the efficiency of the whole process. For obtaining correct and complete evidence, we suggest the final evidence collection should also be carried out by domain experts.

Chapter 7

Functional Safety Management According to ISO 26262

The ISO 26262 standard offers a systematic approach for designing a safe road vehicle (or subsystems of a car) from the design phase through its production. However, providing functional safety (according to ISO 26262) for an existing system is a challenge that needs further effort. This chapter addresses the problem of applying ISO 26262 to a system that has already been developed using conventional safety methods. In this chapter, we provide a methodology for the process of systematically applying the ISO 26262 standard to an existing system. To achieve this, we carried out a case study on an existing autonomous driving system.

7.1 Introduction

Complexity and variety of subsystems in modern vehicles make it cumbersome to design a safe system. A thorough systematic development process is one of the main safety engineering needs [59]. The ISO 26262 standard provides a guideline for addressing the design problem with safety at the center of attention. It also provides a process for developing systems in a car from design to production ensuring its safety. The ISO 26262 standard is based on a V-model process approach. The process starts with the concept phase design where possible vehicle level hazards and safety goals (to mitigate the hazards) are defined. The next step is to define system level requirements based on the safety goals. Afterwards, product development is continued in both hardware and software parts. The following step is the production and operation step. Finally, ISO 26262 recommends the validation and verification through the whole safety lifecycle.

However, the task of applying ISO 26262 to a system that is already designed, or for which the development stage has been completed, requires extra attention. In this chapter we focus on this problem and suggest a methodology for applying ISO 26262 to an existing system. Our approach focuses on reusing parts of the existing system, not tailoring the lifecycle in the ISO 26262. This means all the methods, activities or phases, which are not covered by our procedure but required by the standard, still need to be addressed.

To demonstrate our approach, we carry out a case study on an existing autonomous driving system. Traditionally, some rudimentary safety functions are installed in experimental setups to switch off all automated functions in case the driver or an electronic watchdog detects a mal-functioning of the system. For system evaluation and experimenting by a larger group of drivers as a field operational test, the transition between autonomous driving and manual operation is one of the key aspects. To have an unbiased assessment of the autonomous driving system, the participants involved in the field operational test are not experts. Considering their lack of experience with the system, failure handling needs to be at a safety level in compliance with ISO 26262 [12]. The current study aims to assess how the transition from an experimental autonomous driving setup can evolve towards a sufficient level of functional safety, which allows evaluation of system performance by non-expert test drivers.

This chapter is organized as follows: In Section 7.2 a brief introduction of the relevant parts in the ISO 26262 Part 3 is provided. In Section 7.3 the methodology resulting from our study is presented. Next, the case study that was carried out in the automated driving project is presented in Section 7.4. Moreover, Section 7.5 compares some related articles in this field. Finally, conclusions is presented in Section 7.6.

7.2 ISO 26262 Part 3

In this chapter, the conceptual phase of the proposed process mainly follows the ISO 26262 Part 3. Therefore, in this section, we give a brief introduction to the relevant phases in the ISO 26262 Part 3. Three phases are discussed here: Item Definition, Hazard Analysis and Risk Assessment, and Functional Safety Concept.

Item Definition An "Item" is defined as *a system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied* [12]. An item is usually a system, like a vehicle, a subsystem or large component, etc. But an item can also be a function that is distributed over several systems. In this chapter, the autonomous driving system can be defined as an item. In the item definition document, the functionality of the system is described and the preliminary architecture of the system can be defined. The architectural description could include: the behavior of the system from users' point of view, the internal components of the system and their communication details, the internal process of the system, e.g. the operating states, and the communication of the system under study with other systems in the vehicle. Moreover, a description of known failure modes and potential hazards on the vehicle level is encouraged.

Hazard Analysis and Risk Assessment Hazardous situations are a reality and do occur. It is crucial to identify them and mitigate their occurrence to have a safe design. The Hazard Analysis and Risk Assessment (HARA) method is the procedure in which the designers classify possible hazards according to ISO 26262. In this stage, vehicle level hazards should be determined and documented. Then a set of safety goals are specified for preventing or mitigating a hazard at the vehicle level. During the development of a system, safety goals are refined through a hierarchy of safety requirements. Each safety goal is attributed an Automotive Safety Integrity Level (ASIL) value according to the risk classification of the associated hazard. The ASIL values are ASIL A (lowest), ASIL B, ASIL C, ASIL D (highest), and are assigned by assessing the severity, probability of exposure, and controllability of the given hazards. Severity of a hazard represents the severity of

injuries that an hazardous event can be expected to cause. There are four severity classes: S0 (No Injuries), S1 (Light to moderate injuries), S2 (Severe to life-threatening (survival probable) injuries), S3 (Life-threatening (survival uncertain) to fatal injuries). Exposure of a hazard represents the relative expected frequency of the operational conditions in which the injury can possibly happen. There are five exposure classes: E0 (Incredibly unlikely), E1 (Very low probability), E2 (Low probability), E3 (Medium probability), E4 (High probability). Controllability of a hazard represents the relative likelihood that the driver can act to prevent the injury. There are four controllability classes: C0 (Controllable in general), C1 (Simply controllable), C2 (Normally controllable), C3 (Difficult to control or uncontrollable).

Functional Safety Concept The Functional Safety Concept (FSC) consist of a set of Functional Safety Requirements (FSR) assigned to each component of the system. During this phase, safety goals defined at the HARA stage are refined into functional safety requirements. As an attribute of the safety goal, the ASIL value of the safety goal is inherited by each functional safety requirement.

The ISO 26262 standard provides a detailed methodology, called ASIL decomposition, for tailoring ASIL during the design process. If independent architectural elements exist and safety requirements can be implemented redundantly by these elements, these safety requirements might be assigned a lower ASIL. ASIL decomposition can be applied to functional safety requirements, technical safety requirements, and hardware and software requirements. ASIL determines the qualitative and quantitative aspects of safety requirements to be implemented, and safety activities to be conducted during the safety lifecycle to ensure that the risk is kept at an acceptable level. Therefore, the ASIL decomposition of FSR is useful in the consequent steps defined by ISO 26262, especially the implementation phase.

Moreover, for developing the FSC, knowledge about the functionality of the system is required. This means the functionality of the internal components of the system and their connections as well as the internal (functional) process of the system.

7.3 Methodology

In this section, we propose an ISO 26262 standard-compliant functional safety improvement process. This process consists of five key phases: Review and Plan phase, Conceptual phase, Design phase, Implementation phase, and Verification and Validation phase. Figure 7.1 shows an overview of our methodology for improving the functional safety of an existing system. The Review and Plan phase covers all the activities performed before project inception, such as collecting and reviewing the existing project data, defining and assigning roles, and creating a safety plan. Basically this phase follows the requirements in ISO 26262-Part 2. Then in the Conceptual phase, three steps in ISO 26262-Part 3 will be performed: *Item Definition (C1)*, *Hazard Analysis and Risk Assessment (HARA) (C2)*, and *Functional Safety Concept (FSC) (C3)*. Different from developing a new system, existing data of a system, such as the design model and project description, are also taken into account in this process. The outputs of this phase are Functional Safety Requirements (FSRs). At the start of the Design phase (D1&D2), the initial Technical Safety Requirements (TSRs) need to be extracted or derived from the existing requirements of the system (which were probably used for designing the system in the first place). Then a completeness check for the TSRs will be done according to

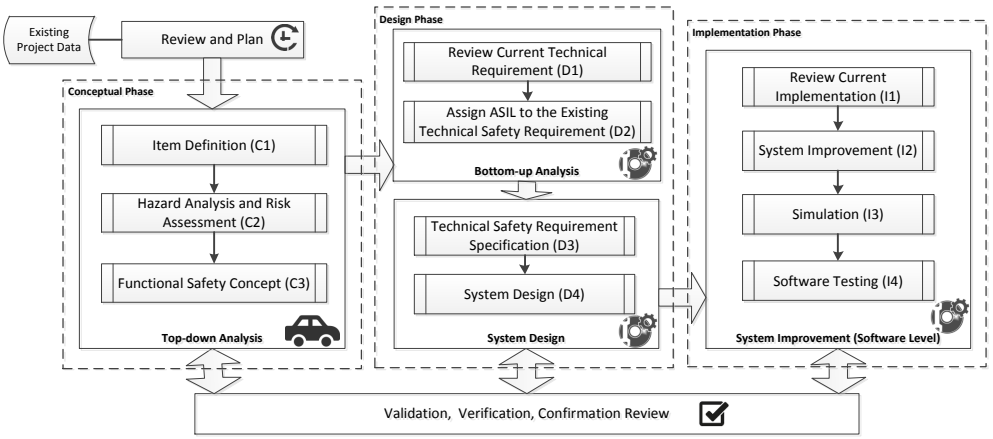


Figure 7.1: An overview of the phases in our methodology for improving functional safety: safety analyses are planned at the Review and Plan phase, and performed during the other four phases.

the FSRs (D3). In this way, the non-implemented TSRs could be identified for future processing. Finally, the TSRs will be used to develop a new system design and the technical safety concept (D4). In the Implementation phase, the existing implementation of the system and the TSRs will be used for improving the system (I1&I2). The updated system will be tested through software simulation and testing procedures (I3&I4). In this chapter, we address only the software development of the system. In parallel to the previous phases, Validation and Verification will be done according to ISO 26262. As an important part of any safety related development process, safety analyses are used through the entire development lifecycle (from the Review and Plan phase to the phases of validation and verification). Because the Review and Plan phase and Verification and Validation (V&V) mainly follow ISO 26262, we postpone the discussion of the related issues in the case study (Section 7.4). We will focus on the other three phases in the following sections.

7.3.1 Conceptual Phase

In this phase, the ultimate goal is to develop functional safety requirements for an existing system. The main workflow of this phase is shown in Figure 7.2. This process mainly follows Part 3 of ISO 26262 (see Chapter 2). To apply this process to an existing system, the function description of the system, which fulfills the requirements in the standard, can be reused for *Item Definition*. Then three fundamental steps are carried out during the HARA: situation analysis and hazard identification, hazard classification, and Automotive Safety Integrity Level (ASIL) determination. Also for an existing system, HARA needs to be done from scratch to extract hazards at the level of the item. A number of well-established techniques can be used for the extraction of hazards, such as checklists, Failure Mode and Effect Analysis (FMEA) [138], Fault Tree Analysis (FTA) [136]. Note that performing safety analyses can be useful in this phase, but is optional. The existing

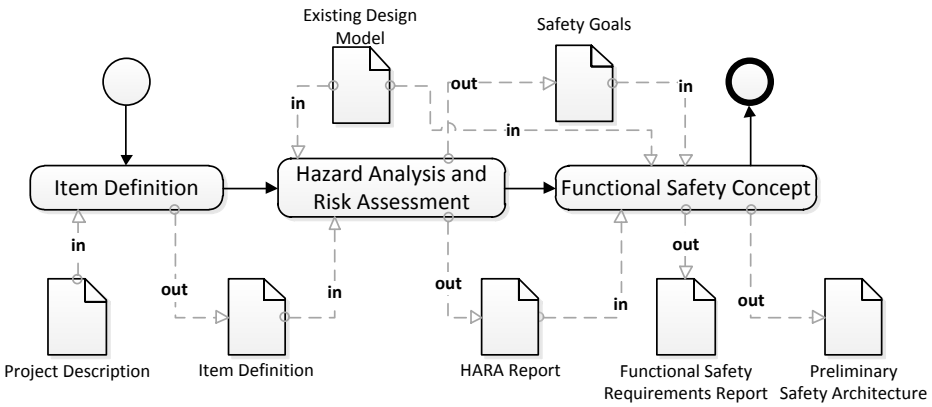


Figure 7.2: The main workflow of the Conceptual phase (only the key activities and work products are shown here).

system architecture design can be used to facilitate the HARA and Functional Safety Concept definition. As a result, a functional safety requirements report and a preliminary safety architecture are created for the target system.

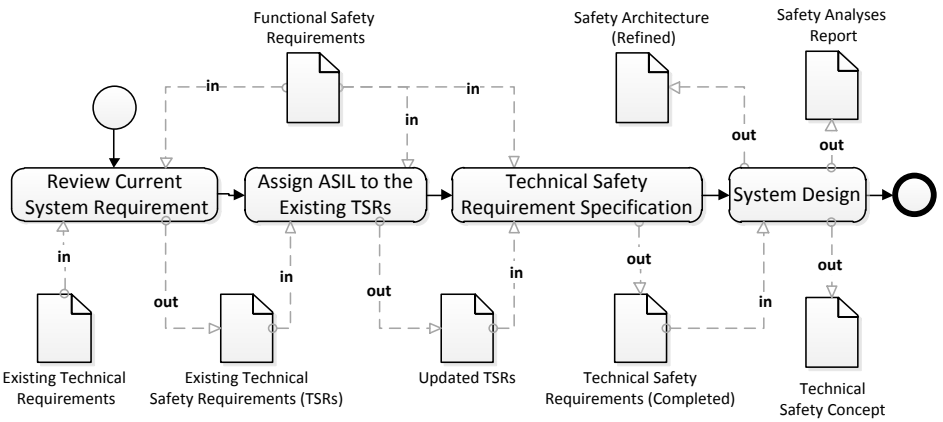


Figure 7.3: The workflow of the design phase (only the key activities and work products are shown here).

7.3.2 Design Phase

In this phase, Sections 4.5–4.7 in ISO 26262-Part 4 are covered. According to the standard, the technical safety requirements are derived from the FSRs that were specified during the Conceptual Phase. However in order to maximize reuse of the existing system data, activities in this phase are re-arranged. Figure 7.3 illustrates our approach to get the TSRs for the existing system. First, a bottom-up analysis is performed by reviewing the system technical requirements. In automotive systems, it can be difficult to isolate safety functions from normal functions. Therefore, by reviewing the initial system, technical requirements and the FSRs, some of the existing TSRs can be extracted and reconstructed. After this, the existing TSRs are linked to the corresponding FSRs. Meanwhile, an ASIL level is assigned to each of the TSRs. Then, a completed version of the TSRs is obtained through a completeness check. The key task here is to guarantee that all the FSRs are developed into the TSRs and to label out the non-implemented TSRs. Finally, the system design and the technical safety concept are developed based on the TSRs specification. Furthermore, safety analyses are used during the system design and then subjected to confirmation reviews. At the system level, safety analyses are required by the standard and the amount and depth of safety analyses are ASIL-dependent. For the lowest ASIL, only a qualitative Failure Mode and Effects Analysis (FMEA) is required. For higher ASIL, Fault Tree Analysis (FTA) and quantitative failure analysis are required in addition. The main outputs of the Design phase are the technical safety concept, a system design description, the refined safety architectural design, and safety analyses reports.

7.3.3 Implementation Phase

This phase concerns the implementation of the technical safety concept at the software level. The relevant part in ISO 26262 is Part 6. Software architectural improvement

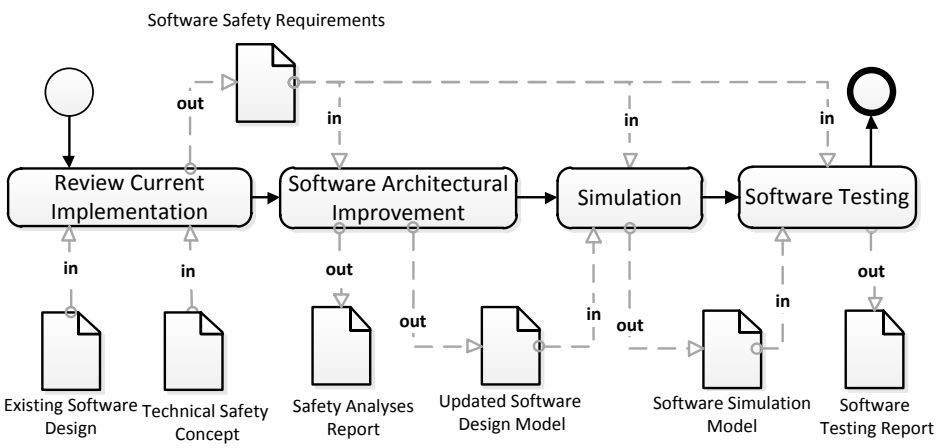


Figure 7.4: The workflow of the implementation phase: only the key activities and work products are shown here.

is carried out according to the non-implemented TSRs and the review of the existing implementation (Figure 7.4). As a result, the software design model gets updated and new safety functions are added to fulfill the non-implemented TSRs. During software design, safety analyses at the software level are carried out. The amount and depth of safety analyses undertaken are also ASIL-dependent. Then, software simulation models are constructed based on the software design model. Finally, software testing can be executed by simulation with a number of test cases, which are created according to the TSRs. The final software testing is carried out during the whole system testing. The output of this phase is a software test report, which specifies the qualification of the software design model with respect to the TSRs.

7.4 Case study

The importance of safety is even more significant in the context of autonomous driving. The performance of automated driving functions is typically limited by the capability of performing scenario assessments. Depending on the assessments of scenarios based on the environment, the speed and position set points are altered. The speed and steering of the vehicle are controlled to reach those set points. Due to the complexity of traffic and other driving conditions, the development of automated driving functions includes experimenting on public roads. The functional performance of autonomous driving can generally be investigated easily using rapid control prototyping systems (e.g. dSPACE [1]). However, for experimenting on public roads an adequate level of functional safety is required. To achieve this, the maturity of the current development process of automated driving needs improvement. In this section, we used an existing autonomous driving system for the case study.

As the existing autonomous driving system is provided by TNO (the largest research institute in the Netherlands), we carried out the case study in cooperation with a number of TNO researchers and developers. In practice, development teams in the research institute typically start developing technologies with attention to basic safety issues only. Many projects are pursued with elementary safety mechanism such as an emergency stop button, or watchdog systems. Specifically for research projects on complex products, such as the automated driving, safety may be compromised to achieve the desired functionality for experiments on a closed track. After achieving the performance and functionality goals, the development teams start to extend on safety aspects that are required for public road use.

In our case study, the development teams in the research institute do not tackle the safety in a systematic way from the design stage. Rather, the safety aspects of the design are integrated in the functionality of the system. Base on this observation, we carry out our research on the effectiveness of a systematic transition towards functional safety based on the ISO 26262 standard. The main goal of this study is to develop methodologies instead of a complete safe product, hence some steps are not discussed in details.

7.4.1 Organization and plan of approach

Our case study is carried out by a research team from Eindhoven University of Technology. The research team consists of eleven PDEng students from the Automotive Systems Design

and Software Technology Program[1]. To approach the functional safety, the research team is divided into two teams: an operation team (six persons) and a validation team (five persons). The operation team is in charge of the system design according to ISO 26262, and work together with TNO researchers who provide support on the technical aspects of the automated driving function. The validation team is in charge of the (internal) verification and validation of the results achieved by the operation team. This team works independently of the operation team. At the end of each step, the results were internally reviewed to assess their compatibility with ISO 26262, and (if required) iterations were done for correcting the work products. Moreover, a validation and verification of the results was carried out by an external consultant company.

The work presented in this section is the outcome of the efforts by the two teams. In this section, the highlights of the results achieved in the process are presented. Details are omitted due to the confidential nature of the research.

7.4.2 Conceptual Phase

7.4.2.1 Item definition

In this stage, contacting the designers of the system was crucial to obtain a clear picture of the system architecture. The development team in TNO provided the operational team the relevant documents, which included the requirements and the current architecture of the existing system. Given the architecture of the system, the operation team defined outlines of the existing system. In addition, adding the context of usage of the system proved useful during the rest stages of the project. Ideally, having a failure mode study included in the item definition can be beneficial for Hazard Analysis and Risk Assessment (HARA) (see Section 7.4.2.2). Nevertheless, obtaining a detailed failure mode analysis is a quite demanding task; due to time restrictions, the project was continued without a failure mode analysis included in the item definition. Finally, going through revision rounds with the designers of the system, the validation team ensured the correctness of the item definition.

One of the most important lessons learned during the project was that it is very important to make sure that the item definition is completed before moving to the HARA, since any change at this level propagates through all further stages.

7.4.2.2 Hazard Analysis and Risk Assessment

Brainstorming methods [127] were used for identifying hazards. The project team benefited from TNO experts input directly in the brainstorming sessions to accelerate the process. In this stage, knowledge about the behavior of the system was exchanged (both from user's perspective, and internal process of the system). A total of 14 different hazards were identified for the system under study. An example of a hazard is shown in Table 7.1. In this table, the rows have the following meaning: *Scenario* is the situation in which the hazard is happening. *Environmental condition*, as the name suggests, is the conditions of the environment for that scenario. *Hazard* is the actual hazard that may happen.[2] *Effect of failure* is what happens in case the hazard takes place. The rest of the rows are related to the ASIL determination according to ISO 26262 (see Section 7.2).

[1]The PDEng programs are two-year salaried programs in the field of technological design. The programs lead to a Professional Doctorate in Engineering degree.

[2]The details of the hazards are omitted due to confidentiality issues.

Table 7.1: An example of a hazard developed during HARA.

Scenario	Normal driving on a straight highway, speed > 30 km/h
Environmental condition	Slippery road, with lead vehicle in front
Hazard	—
Effect of failure	A collision might happen. Vehicle might spin out of control
Severity classification	S 3
Justification	Rear/front end collision with lead vehicle or other car at medium speed
Exposure classification	E 3
Justification	Driving on slippery (E 3 for wet roads, and E 2 for snow/ice) highway
Controllability classification	C 3
Justification	C 3: Car spins out of control, C 2: Snow and icy road, C 1: Wet road
ASIL	ASIL C

7.4.2.3 Functional Safety Concept

According to ISO 26262, after HARA, the next step is to develop a Functional Safety Concept (FSC) for the system. In this stage the operation team started with a preliminary architecture based on the existing implementation of the function. After defining the Functional Safety Requirements (FSR) for each component and assigning them to each component of the system, a proper ASIL value was assigned to each safety requirement. The operational team applied ASIL decomposition to functional safety requirements. However, it was assessed by the operational team and the TNO development team that within the existing architecture a significant decrease in the ASIL value could not be achieved using decomposition rules. To be able to reduce the ASIL value, some minor changes were proposed by the operational team for the system architecture.

In the FSC stage already the benefits of following ISO 26262 methodology were identified, for instance, an optimal architecture with respect to safety of the system is difficult to achieve without an insight in the safety critical components and their impacts on each other.

7.4.3 Design phase: Technical Safety Concept

Definition of the Technical Safety Concept (TSC) is the next stage. In this step, according to ISO 26262, technical safety requirements were developed with respect to the functional safety requirements acquired in the previous step. However, since the system already existed and was designed according to a set of requirements, an additional step is needed for integrating the original requirements (which may be a mixture of (non)functional requirements, and functional *safety* requirements) with the new set of TSRs. The original requirements of the autonomous driving system were analyzed to identify safety related requirements. These requirements were then merged into the TSR. Moreover, proper ASIL value has been assigned to each safety requirement of a component of the system. During this phase, the operational team also applied the ASIL decomposition to TSR, consequently, some changes were made to the system design to fulfill the decomposition rules.

7.4.4 Implementation phase: Developing models and test results

After developing the technical safety concept, the changes in the existing architecture of the autonomous driving system were implemented in simulation models. Then, appropriate safety mechanisms were developed according to each technical safety requirement. This step was quite straightforward (in comparison to the previous steps) as the requirements were clear and already allocated to components. After having all the technical safety requirements implemented in the autonomous driving system, a functionality test was performed on the system to ensure that the changes in the safety did not affect the nominal functionality and performance of the system design.

The implementation results were tested internally (by the validation team) and feedback on the safety level was provided. Not every safety aspect was addressed with the first revision of the system design and a second round of revision was performed. However, the second revision was finished faster than the first, because the traceability of each requirement facilitates the analysis process of the reported bugs.

7.4.5 Internal Verification and Validation

7.4.5.1 Plan of approach

According to ISO 26262 each work product such as HARA, FSC, and TSC, has to be verified. In this case study, the validation team designed a number of checklists according to the standard. then these checklist were used for the Internal Verification and Validation (V&V). Moreover, the validation team also developed a number of test cases to verify the functionality of the implemented models.

7.4.5.2 Conceptual phase

During conceptual phase, there are three work products of the autonomous driving system delivered to the validation team: Hazard Analysis and Risk Analysis report, Functional Safety Concept report, and Technical Safety Concept report. According to the ISO 26262 standard, the validation team carried out the Verification and Validation as an iterative process. Revisions were carried out until all the parties were convinced that nothing is missing from the documents.

HARA The HARA report of the autonomous driving system was reviewed based on questions from the checklist with the project interpretation. The report consists of findings, problems and explanations, which clearly state possible inconsistencies with ISO 26262 and suggestions for possible improvements.

One of the important lessons learned during this project was that it is of value to perform many iterations between the operation and verification teams to ensure the quality of results. This is an important procedure to achieve satisfactory and correct results. In case this procedure is not performed correctly, it may create incompatibility issues in the next stages of a safety design.

FSC The V&V is performed on FSC in the same manner as in case of HARA. The checklist of functional safety requirements was developed according to the requirements in the ISO 26262 Part 3. The checking process consists of the compliance checking with ISO 26262 and project interpretation, the completeness and consistency checking on safety

requirements, and the correctness checking on ASIL decomposition. As an output of this stage a verification report was delivered by the validation team.

7.4.5.3 Design phase

TSC The next step was dedicated to the identification of technical safety requirements that could be analyzed at the design level and could ensure fulfillment of functional safety requirements.

Similar to previous stages, a new checklist for technical safety requirements with the project interpretation was developed. The checklist of TSC deals with the following concerns: compliance and consistency with FSC, compliance with preliminary architecture design assumptions, and the correctness checking on ASIL decomposition. The checklist was derived based on ISO 26262-1:2011, ISO 26262-4:2011, and ISO 26262-8:2011. The results of this validation phase were documented in a validation report on TSC.

7.4.5.4 Implementation phase

To carry out V&V on the functional safety models, several pre-requisites have to be met according to the standard. Technical safety requirements have to be determined, approved, and implemented to go forward with the verification. To address this, the implementation of the safety requirements were suggested to proceed after the validation of TSR. In this chapter, based on the safety goals, several test cases with different scenarios were developed, documented, and executed.

Safety test cases As mentioned before the verification team designed the safety test cases for testing the implementation of the system. Test cases are scenarios that have to be translated into a code which inputs the fault or disturbances into the model. Based on the input from each test scenario, results of testing are obtained. These results show whether the implementation fulfills expected output requirements, which ensure the safe functioning of the system. According to ISO 26262, these safety test cases were developed based on TSR.

Due to the time limited of the project, the validation team designed ten test cases according to the safety requirements. All test cases were developed in the Matlab/Simulink environment. Test scenarios assume usage of the model provided by the operation team. The main idea of designing scenarios is to include faults by adding fault injection boxes in the main layer of the model. Finally, the report was generated which clearly states the PASS or FAIL message related to each test scenario. The validation team used the same set of test cases to test the original design of the system and the new design of the system. The results show that before applying ISO 26262, only four out of these ten test cases passed, however for the new design, eight of these ten test cases passed. The reason of two failed test cases is the design of the system does not cover the scenarios assumed by these two test cases. Although test cases and design of the system are developed based on the same set of safety requirements, they are carried out by two independent teams and each team has their own interpretation of these safety requirements. Therefore, the gaps between system design and test cases might exist. To address this, the manager of the project needs to decide whether the scenarios assumed by these two test cases are not important or the design of the system should be improved to address these test cases. However, based on the simulation results, we conclude that our approach can facilitate the improvement of the system architecture design.

7.4.6 External Verification and Validation

Although Verification and Validation are frequently combined, in our case study the external V&V, which is performed by a third-party company, only focuses on Validation. Validation means to investigate whether the system has been constructed according to its specification. Verification, meaning to investigate whether the system has been made in the correct way, is assumed to be executed in each of the previous phases. The external validation in our case study is built up of two parts: one part describing the preconditions of the existing system, and one part describing the validation activities of the system with safety improvements. It is the last phase that we are able to prove the required safety. The external validation of the improved system in the case study mainly focuses on:

- A thorough analysis of the system is performed: FMEA, FTA and System Safety FMEA or Failure Mode and Effect Diagnostic Analysis (FMEDA).
- Establish or update the tracing of all requirements, now including the FSRs and TSRs, towards validation. In this way, we can ensure the full coverage of validation.
- Test reports have been evaluated and concluded a 100% coverage of all safety requirements, or the residual risk of the tests on safety requirements have been accepted by the Project and Safety Management. Test cases need to be extended, e.g. in safety systems, injection of failures is common practice.
- Independent confirmation measures, audit and assessment have been performed and reported on the phases, ensuring the defined safety process has been followed, and the process complies to ISO 26262. After the execution of safety analysis, acceptance of the tests and the confirmation of ISO 26262 compliance, the final conclusion of the improvement project can be made.

For reasons of confidentiality, the details of the external validation of the case study are not discussed here. However, the external validation results show that our approach are ISO 26262 compliant.

7.5 Related work

This chapter presents a process of system development, which complies with ISO 26262. There exist several similar approaches. Siemens Drive Technology Division uses ISO 26262-compliant and a model-based development process to increase safety-critical vehicle functions [43]. However this work considers only the software level. Our work concentrates on integration of the safety standard for hardware and software into automotive systems. Mirko Conrad [53] proposed to employ a workflow for verification and validation of models and generated code. It follows the validation and verification requirements outlined in ISO 26262-6, but this workflow only focuses on software development. Ward and Crozier [147] describe the ASIL decomposition that is applied to meet customer requirements. The proposed approach does not evaluate the methodology for improving functional safety. However, our approach is based on the process that was experienced during developing a safety system for the automotive developer. The case study shows that our approach can be used for improving functional safety of an existing system.

Moreover, recently some research focuses on using model-driven techniques to facilitate safety engineering according to ISO 26262. A semi-formal safety engineering approach based on SysML was proposed for specifying the relevant safety artifacts in the concept

phase of ISO 26262 [103]. The approach was designed to improve the common understanding of the relevant safety aspects during the system design. Besides, a structural modeling and annotation of failure data was developed for hardware architectural designs [32]. The goal of this study is to support a rapid quantitative safety analysis regarding evaluation of the hardware architectural metrics and evaluation of safety goal violations.

7.6 Conclusion

In this chapter we introduced a methodology to improve functional safety for an existing automotive system, according to ISO 26262. It includes five main phases: Review and Plan phase, Conceptual phase, Design phase, Implementation phase, and Verification and Validation. Our main goals is to apply ISO 26262 to an existing system and to maximize the reuse of the existing project data. An industrial case study has been used to illustrate our approach in a detailed way. The validation results show that for the original design of the system, only four out of these ten test cases are passed, however for the new design, eight of these ten test cases are passed. Based on this, we conclude that our approach can facilitate the improvement of the system architecture design. Finally, an external validation by domain experts has been performed for our case study.

Chapter 8

Safety Metrics Design for Safety Assurance

In the safety domain, safety assessment is used to show that safety-critical systems meet the required safety objectives. This process is also referred to as safety assurance and certification. During this procedure, safety standards are used as development guidelines to keep the risk at an acceptable level. Safety-critical systems can be assessed according to those safety standards. Due to the manual work involved, safety assessment processes are costly, time consuming, and hard to estimate. The goal of this chapter is to design metrics for safety assessment. These metrics can, for instance, identify costly processes in the safety assessment process. In this chapter we propose a methodology to design metrics for safety assessment from different perspectives.

Metrics can be identified by answering three questions. Three different sources of information have been identified for obtaining metrics: industrial interests, safety standards, and available data. For each of these sources appropriate methods have been proposed and used for obtaining the relevant metrics. A case study in the context of the European project OPENCOSS is carried out to demonstrate the method. Finally, a validation of the obtained relevant metrics has been done by means of a survey amongst 24 experts from 13 project partners. Different methods for designing metrics have to be used for each source. The validation shows that most of the relevant metrics are useful for industry.

8.1 Introduction

Safety standards, such as ISO 26262 [12], are proposed to guarantee safety risks at an acceptable level. Safety engineers in the safety domains (such as automotive, avionic, and railway) manually check the development process of safety-critical systems for compliance with the standards. This checking process is referred to as safety assessment or certification. Due to the huge amount of manual work involved, safety assessment is costly and time-consuming [57]. Metrics, such as the time spent on the safety assessment process, can be used to estimate the overall cost and monitor the whole compliance process. They can also help to identify the costly activities related to safety assessment. However, if incorrect information is identified, wrong decisions can be made. If unnecessary data is collected,

it will increase the cost, effort, and reduce the effectiveness [40]. Furthermore, important aspects cannot be analyzed if data is missing. Thus, we should not only consider what metrics for safety assessment can be designed, but also what data is available to be measured.

A number of methodologies have been proposed for designing metrics. Goal Question Metric (GQM) is a common approach for metric design [41]. It first defines an objective, then refines it into questions. Finally metrics are derived to answer those questions. Based on GQM, a software measurement framework, called Practical Software and Systems Measurement (PSM) [105], has been developed. According to industrial practices and experience, PSM provides a guideline and suggestions for implementing a software measurement program. Moreover, a method to alter PSM to include safety is proposed [109]. This method provides two approaches (top-down approach and bottom-up approach) for designing metrics. However, both of them have disadvantages. For the top-down approach, it does not consider the feasibility of base measures, for example, whether suitable measurable entities actually exists in an area of work. For the bottom-up approach, it does not consider the purpose of measurements. When using this method for designing safety measures, a balance between these two approaches should be found. In this chapter, based on the different perspectives (industrial interests, safety standards, available data), we formulate three questions (*What do we want to measure? What should we measure? What can we measure?*) for designing metrics for safety assessment. Then different approaches are applied to get answers to these questions. Finally a number of metrics for safety assessment can be obtained and implemented. For demonstration of our process, we carried out a case study in the context of an FP7 European project (OPENCOSS [61]). The case study focuses on safety assessment in the automotive domain (ISO 26262) regarding functional safety.

The rest of the chapter is organized as follows: Section 8.2 discusses the Practical Software and Systems Measurement framework. Section 8.3 provides the research methodology used in this chapter. In Section 8.4 we discuss questions for the metric design, as well as methods for getting the answers. Section 8.5 introduces a case study in the context of the OPENCOSS project. Then, the validation of the final results is discussed in Section 8.6. Section 8.7 shows the related work in this field. Finally, concluding remarks and future work are presented in Section 8.8.

8.2 Practical Software and Systems Measurement

The *Goal Question Metric* is a data collection method for evaluating software development methodologies and improving the software development process [41]. It can be used for understanding the fundamentals of measurements, identifying information needs, and defining measurement goals. The GQM is a top-down approach and uses goal-directed data collection. It starts with a set of corporate, division and/or project business goals, then derives questions for achieving these goals, and finally it identifies the metrics answering the questions.

PSM is based on the *Goal Question Metric* approach, and standardized in ISO/IEC 15939 [81]. This process encourages [109]:

1. the identification of Information Needs (IN);
2. the interpretation of an Information Need as being within an Information Category (Cat);

3. the identification of Measurable Concepts (MC) within each Information Category;
4. the identification of Prospective Measures (PM), associated with each Measurable Concept.

One of the key contributions of the PSM framework is the Information Category-Measurable Concept-Prospective Measures (ICM) Table [16]. This table contains a categorization of concerns, and base measures which can be used to address the corresponding concerns. The mapping between Information Categories, Measurable Concepts and Prospective Measures is recorded in the ICM table. When using the ICM table, a user could follow the recommended process in the PSM to identify their Information Needs, select Prospective Measures, and map these measures to the available artifacts. Besides, an extension of the ICM table for the system safety domain is proposed by Murdoch [109].

8.3 Research Methodology

In this section, we discuss the research methodology used in this chapter. There are five steps in our research methodology: define the central research question, decompose the central research question, propose methods to address research question, apply the methods to a case study, validation of the results, and discuss the threats to validity.

Step 1: Define the central research question Before carrying out the research on safety assessment metrics, the goal of the research should be clearly defined as the central research question. In this chapter, the central research question is as follows: *Which metrics are relevant for safety assessment process?*

Step 2: Decompose the central research question The central research question is split into a number of more specific research questions. Each of these specific ones can be addressed separately. In this chapter, there are three specific research questions derived from the center research question: *What do we want to measure? What should we measure? What can we measure?*

Step 3: Propose methods to address research question After formulating the research questions, the methods to address these questions should be found out. These methods can be based on existing techniques or new proposed ones, for instance, for extracting the metrics from the ISO 26262 standard, we propose a method based on the existing PSM framework. In this chapter, different methods are proposed for the individual research question. The details of this are discussed in the Section 8.4.

Step 4: Apply the methods to a case study To demonstrate how to apply the methods, a case study has to be carried out. The reporting of the case study should be given, which includes the description of the case study as well as the representation, interpretation and discussion of the results [89] [129]. In this chapter, we perform a case study in the context of the FP7 project OPENCOSS. For designing standard-related metrics, we select the ISO 26262 standard in the automotive domain. The details of this case study are described in the Section 8.5.

Step 5: Validation of the results After obtaining the relevant metrics, a validation is done to evaluate usefulness of the metrics. In this chapter, we carry out a survey of 24 experts from the 13 project partners. The details of this validation is provided in the Section 8.6.

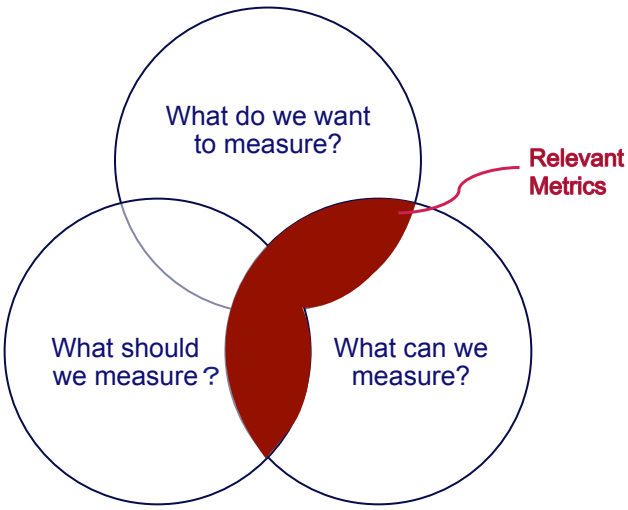


Figure 8.1: Metric design for safety assessment.

Step 6: Discuss the threats to validity Finally the threats to validity of the proposed methods are discussed. For our methods, the threats to validity are given in the Section 8.6.

8.4 Design Metrics for Safety Assessment

When designing metrics for safety assessment, there are three questions to be answered. What do we want to measure? What should we measure? What can we measure? We propose to derive the relevant metrics from the answers of those three questions (Figure 8.1). For safety assessment, manufacturers, who produce safety-critical systems in the safety domain, have their own interests in metrics. We assume that the manufacturers always know what they want to measure. Thus the answer for the first question is based on industrial interests or needs. Moreover, safety assessment is always carried out according to safety standards. Those safety standards are used as guidelines for system development. Therefore metrics designed according to safety standards are the answer to the second question. For the last question, it can be answered based on the available data. Metrics can be designed before the data gets collected. However, the metrics can only be applied in a project when there are data available to be measured. For example, if a manufacturer wants to measure the number of system hazards, but if the data on system hazards is not available, this metric can not be used. The relevant metrics are the highlighted part in Figure 8.1. If the available data (What can we measure?) is not taken into consideration, the ideal situation is that all the metrics derived from the first two questions (What do we want to measure? and What should we measure?) are all relevant metrics. But the metrics can only be used when there is data available. Therefore, the relevant metrics consist of the intersection of *what we can measure* with *what we want to measure*, unified with the intersection of *what we can measure* with *what we should measure*.

Based on this observation, we identify three sources for safety assessment metrics:

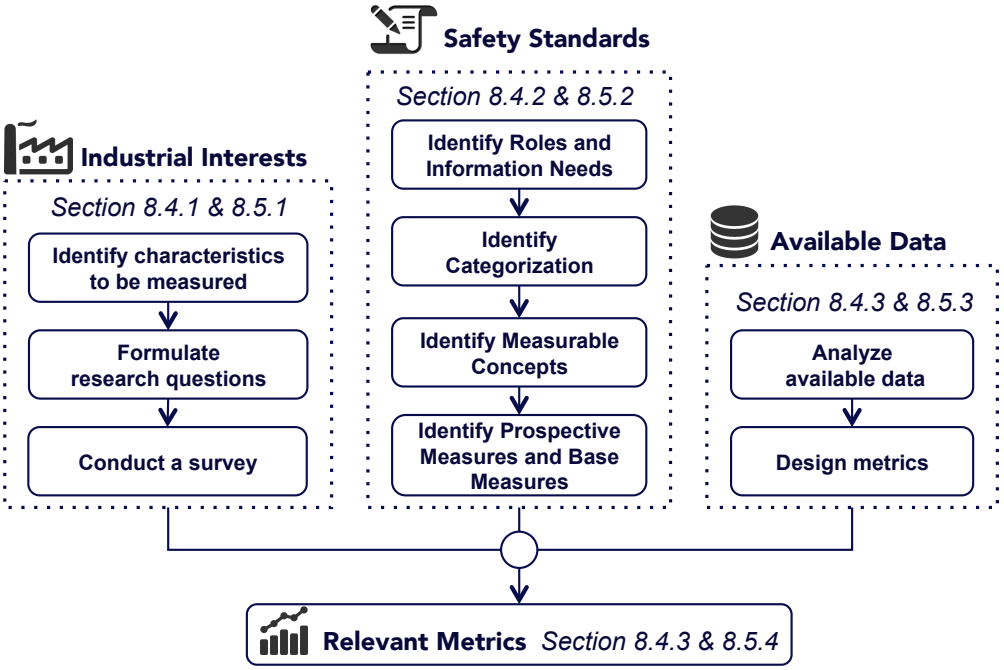


Figure 8.2: An overview of the process for designing safety assessment metrics.

industrial interests, safety standards, available data. The methods for designing metrics from those sources are discussed in the following sections. Figure 8.2 shows an overview of the process for designing safety assessment metrics. For different sources, different methods are applied. A survey is used to get metrics based on industrial interests. An adaptation of PSM is proposed for designing metrics according to the safety standards. A bottom-up approach is used to design metrics based on available data. Finally, relevant metrics can be obtained by combining the metrics from these three sources.

8.4.1 What do we want to measure? Industrial Interests

Manufacturers are strongly involved in the process of safety assessment for their products. They want to assess the status of their project to see if it is on track or in need of process improvement. Normally and hopefully, they have experience and knowledge of safety assessment. Therefore, their interests should be taken into consideration when designing metrics. There are various ways to get metrics based on industrial interests: questionnaires, brainstorming, project meetings, conferences, interviews. We propose to use a questionnaire-based survey to get information from industry. Surveys can generalize empirical findings to many projects, get benefits from existing experience, and help to identify best/worst practices [69] [67] [58]. Based on GQM, we summarize the typical steps for a survey research [67] [68] [75] into three main steps: identify characteristics to be measured on the product and process, formulate research questions for the survey, and conduct the survey. Finally the metrics based on industrial interests can be obtained through the survey or be derived from the results of the survey.

8.4.1.1 Identify characteristics to be measured

In this phase, the characteristics that are related to safety assessment should be identified. The characteristics represent the interests of the survey and can be used for designing research questions. Typically, two types of characteristics can be identified to be used to estimate assessment efforts or to measure coverage of safety assessment goals.

Product characteristics: the elements that characterize the work products created during the safety assessment process. They can be used to identify the information needs for the work products. For example, based on the work product *Functional Safety Requirements Report*, an information need could be number and complexity of functional safety requirements.

Process characteristics: the elements that characterize the development process of safety-critical systems or safety assessment process. They can be used to identify the information needs for the certification process or project-specific process, for example, the cost or progress of the assessment process.

8.4.1.2 Formulate research questions

Based on the result of the previous step, the research questions can be formulated. Clear and explicit research questions show the goal of a survey. They can be further used to facilitate the questions design in the survey. For example, from process perspective, a research question could be proposed: "What metrics do practitioners need for planning and/or monitoring the development of a safety critical system?" The objective of this research question is to evaluate the status of the system development process.

To address this question, questions in the survey can be derived using the existing methods on survey question design [66] [67]. For instance, the following questions in the survey can be derived from the previous research question: "Is the cost of the development of a safety-critical system measured? If yes, how is it measured?", "What aspects do you think should be measured to increase the transparency of the system development?" Moreover, after collecting survey data, these research questions can be answered through data analysis.

8.4.1.3 Conduct a survey

After formulating research questions, the survey can be carried out. As mentioned before, a survey can produce data based on real-world observations [85]. It is an empirical method for collecting information from experts. To conduct a survey, five steps can be followed [67]: design of the survey schema, validation of the survey schema, data collection, data analysis, and reporting.

The survey schema in form of a questionnaire is designed to ask respondents (domain experts) to provide information about their interests, knowledge and current practices in relation to safety assessment metrics. Moreover the questions in the survey schema are derived from the aforementioned research questions. Then a validation of the schema is carried out to get feedback on the understandability and reliability of the questionnaire, as well as the time estimation of filling the questionnaire. Modification or refinement of the questionnaire might be necessary. Then the questionnaire will be filled in by the respondents. After that, the survey data is analyzed to answer the original research questions formulated in Section 8.4.1.2. Finally, the results and conclusions of the survey

can be documented in a report. The metrics derived from the survey represent the industrial interests on safety assessment metrics.

8.4.2 What should we measure? Safety Standards

As compliance with safety standards is one of the crucial parts of safety assessment, safety standards should be taken into consideration when designing safety assessment metrics. Those safety standard metrics can show the degree of compliance with safety standard, and can be used to monitor the safety assessment process. Therefore, we propose to extract metrics from safety standards to support safety assessment process. A method based on PSM has been proposed by Murdoch for introducing safety concepts into the PSM [109] [108]. This method provides two approaches for designing metrics: a top-down approach and a bottom-up approach. For designing metrics according to safety standards, a balance between these two approaches should be found. Therefore, we adapt these two approaches into one procedure. In this way, we consider not only the information need of roles, but also the requirements, activities, and work products in the safety standards. This adapted procedure consists of the following steps:

Step 1: Identify Roles involved in the standard compliance process, e.g., program manager, system engineer.

Step 2: Consider their Information Needs based on the requirements in the standard, then assign a Category to these Information Needs. If an Information Need can not be put into the existing Category in the ICM table, a new Category can be introduced.

Step 3: Identify Measurable Concepts that are candidates for meeting the Information Needs. If an Information Need can be assigned to an existing Category, the corresponding Measurable Concepts in that Category will be checked. If necessary, new Measurable Concepts can be introduced.

Step 4: Identify relevant Prospective Measures and Base Measures according to the Measurable Concepts. The Prospective Measures and Base Measures can be derived

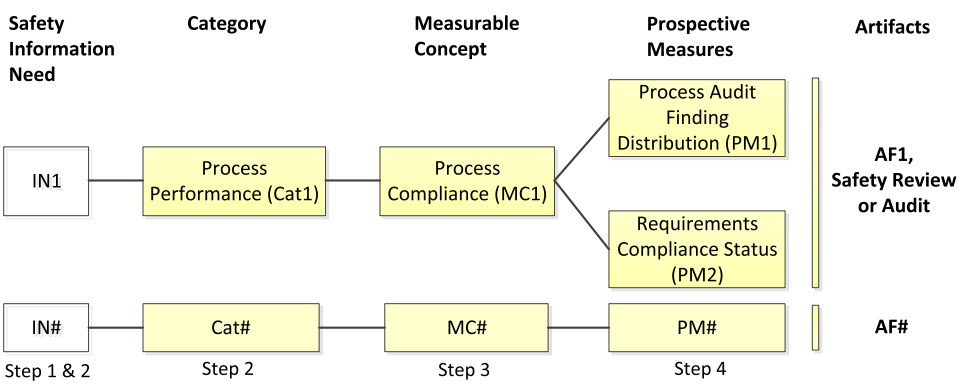


Figure 8.3: Measurement construct in PSM based on [109, Figure 1].

from activities, work products and their attributes in the safety standard. If necessary, new Prospective Measures and Base Measures can be added.

Step 5: Summarize the modifications to the ICM table.

We could see that the adapted appraoch is still a top-down approach, but it also ensure that base measures are driven by the actual entities and attributes in the safety standard. In this way, the existing data in the standard would be used as much as possible. In this chapter, we discuss how to adapt the PSM for the automotive domain (ISO 26262 standard). We selected one concrete example from our case study (Section 8.5) to demonstrate the adaptation process. The results of this example are presented in Figure 8.3.

Roles and Information Needs The Roles relevant to safety assessment are identified in the first step. To do so, a number of typical Roles are identified, which are applicable to the safety domain, such as specialty engineer, system engineer, acquirer, developer, end user/operator, supplier, and regulator. For each Role, typical Information Needs are discussed by Murdoch [108]. This is done by analyzing the responsibilities of the Role.

However, depending on the context, some other Roles can also be identified and their responsibilities should be specified. For example, for safety assessment, a Role called program manager is identified for managing the safety standard compliance process. According to the requirements in the item definition (ISO 26262 Part 3 Section 5), The following concerns can be considered: *"How many functional requirements are covered in the operating scenarios?" "What is the degree of confidence that the assumptions made on the environment are correct and sound?" "Are all potential consequences of an item described?"* These concerns can be merged into one general concern: *"How well does the work comply with the standard requirements on the process?"* (Figure 8.3 IN1).

Categorization Categorization of the Information Needs is the next step. The goal of this step is to interpret and categorize the Information Needs. This is concluded by categorizing the need in an existing category in the ICM table or by defining a new category. The Information Need IN1 in Figure 8.3 is categorized as *Process Performance* (Figure 8.3 Cat1).

Identifying Measurable Concepts Identifying Measurable Concepts is the third step. "A Measurable Concept is an abstract relationship between attributes of entities and Information Needs." (IEC 15939). The Measurable Concept is used for satisfying Information Needs, and can be identified by using relevant entities and their attributes. These concepts can be derived from the ICM table or a new added one. In the example, the relevant entity in the ISO 26262 is the report of item definition. The Measurable Concept should describe the relation between attributes of this report and the aforementioned Information Needs. Therefore, the existing Measurable Concept *Process Compliance* in the ICM table is selected.

Prospective Measures and Base Measures Prospective Measures are identified in the last step. A Prospective Measure is used as a guide to implement an actual Base Measure. This can be done in terms of one or more attributes of artifacts. Examples of these are concrete work products or other existing or introduced entities existing within a project. For our example two Prospective Measures are identified: Process Audit Findings

Distribution (Figure 8.3 PM1) and Requirements Compliance Status (Figure 8.3 PM2). Both of these two Prospective Measures can be derived from the Safety Review or Safety Audit activities in the ISO 26262. Note that, the Requirements Compliance Status is a new Prospective Measure. Then based on the Prospective Measures and relevant activities in the standard, some Base Measures are recommended (see Section 8.5.2.3). Most of these measures could be derived or implemented from the reports of *Safety Review* or *Audit* (Figure 8.3 AF1).

8.4.3 What can we measure? Available data

We would like to gain benefits from available data as much as possible. When designing metrics, available data has to be considered. A huge amount of metrics can be defined or derived from different sources, for instance, safety standards and industrial interests. But we can only base the measurements on the available data for a given project or in some database. If the available data leads to additional constraints for the metrics, some metrics will be eliminated.

A number of methods can be used in this phase, such as brainstorming, conducting surveys, or having project meetings, in order to decide which metrics are relevant. Different from the GQM approach, the process of this phase is bottom-up. What kind of metrics can be derived from existing data and which goals can be achieved through these derived metrics, can be established via analysis of the data. The target metrics can be selected based on the industrial interests or safety standard requirements.

Finally, this phase could have effects on the data collection. If some metrics are highly relevant and crucial but the data is missing, thus the data for these metrics should be collected and stored.

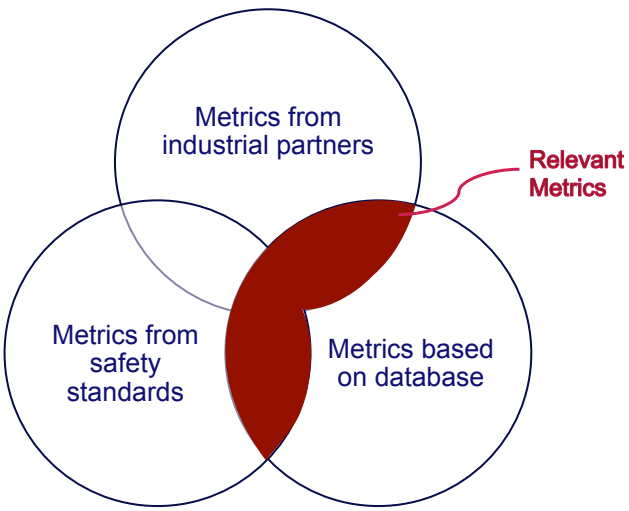


Figure 8.4: Answers for metric design questions.

8.5 Case Study

A case study has been performed in the context of the FP7 project OPENCOSS [61] to demonstrate our approach. The goal of this OPENCOSS project is to establish a common certification framework for different safety domains. As a result, an open-source safety certification infrastructure based on a common certification language (CCL) has been realized [23]. The CCL is defined to improve the mutual recognition of safety assessment from different safety domains, such as automotive, railway, and avionics. It consists of a collection of metamodels to describe the various aspects of safety certification. In the OPENCOSS infrastructure, the CCL serves as an enabling technology for the approaches to model compositional argumentation, evidence management and process compliance. All the data created or used in the OPENCOSS infrastructure is stored in a database, which is derived from the CCL metamodels. Evidence on safety assessment can be stored in the infrastructure. Documents can be used as evidence, such as hazard analysis and risk assessment report. One of the main tasks of this infrastructure is to support transparent certification, which means that the process for certifying components or elements is clearly defined and it can be measured and traced. Metrics are required to be designed and implemented in the safety certification infrastructure.

For our case study, the answers for the three design questions (see Section 8.4) are shown in Figure 8.4. In the following sections, we discuss these three parts in details.

8.5.1 Industrial Interests

In OPENCOSS, different approaches were used to collect industrial interests for safety assessment metrics. Those approaches included project meetings, literature review, and a survey. Finally 168 metrics were obtained from the industrial partners. These metrics consist of 108 metrics from the survey and 60 metrics from project meetings and industrial-relevant literature review. These 60 metrics are obtained in a non-structural way, such as mentioned during project meetings, personal communication with project members, research on industrial-relevant publications on metric. For other metrics, researchers from eight project partners are involved in the design of the survey, and domain experts from five industrial partners have participated in the survey. Details of the survey metrics based on the industrial interests can be found in OPENCOSS deliverable D7.4 [23]. For example, there is a metric from the survey called "number of software hazards", this metric is intended for measuring "hazards that contains one or more software causes", the expected value of this metric is "a real number or percentage of total hazards".

8.5.2 Safety Standards: ISO 26262

In this section, we use Part 3 of ISO 26262 standard as an example. The goal of this part of the case study is to demonstrate how to derive metrics from the ISO 26262 standard using the PSM framework. To achieve this, some alterations to the ICM Table are made. The details of these additions are explained in Section 8.5.2.3. In the following sections, each identified information need is denoted by *IN* + a number. Also for each of the proposed Information Needs, the relevant sections in the standard ISO 26262 Part 3 are referenced. The following referencing schema will be used: *ISO:section within the standard*. For example ISO:5.4.1.a refers to item *a* in section 5.4.1 of ISO 26262 Part 3.

ISO 26262 Part 3 consists of eight clauses. The first four clauses [ISO:3.1-3.4] explain the applicability of the standard and the terminology used in the standard. Therefore,

these clauses are not included in this case study. Each of the remaining clauses has a set of objectives and requirements to be met. Clauses 5 and 6, *Item definition* and *Initiation of the safety lifecycle* [ISO:3.5-3.6] (re)define an item. These two clauses mainly describe requirements of the required work products. In the last two clauses, *Hazard analysis and risk assessment* and *Functional safety concept* [ISO:3.7-3.8], an analysis is done on an item. The requirements presented in these two clauses mainly focus on the process.

Roles The ISO 26262 does not contain the information of roles, however, when a company applies the standard to their projects, a number of roles will be involved. These roles have a set of responsibilities which derive their needs for information. Based on the objectives and the requirements defined in ISO 26262, we distinguish two Roles throughout this case study. In different contexts, different roles might be identified. The first Role is the business manager, who is responsible for all financial costs of a project. Their primary concerns are resources and the cost of those resources. The standard does not include the financial management, however it does describe situations which could influence the costs, such as rework required for modifying an existing item, or legal requirements to be compliant with. For our case study, those situations are identified. The second Role is the program manager. One of the key responsibilities of a program manager is to manage the compliance process according to the standard. They need to provide evidence for safety argumentation.

8.5.2.1 Information Need

For each clause from ISO 26262 Part 3, we identify Information Needs for each Role. Based on our knowledge, for the business manager, there is a general concern: what is the time spend on a project? This derives the first Information Need:

IN 1: What are the staff costs of the entire process?

Item definition The main objective of Item Definition [ISO:5] is to define and describe an item, its dependencies on, and its interactions with, the environment and other items. Another objective is to support adequate understanding of an item for further development. To achieve these objectives, requirements are derived for the item [ISO:5.4.1] and its boundaries [ISO:5.4.2].

The basic concern of a business manager is the cost of compliance. The requirements for Item Definition include legal requirements [ISO:5.4.1.c], which always impose extra costs, other than developing costs, for licensing and certification. The concerns derived from these requirements can be summarized into:

IN 2: What are the overhead costs when developing an item?

For a program manager, the concerns focus on the compliance process and time management. They need to check if all the requirements in the standard are covered by their *Item Definition*. Thus they have the following concerns:

- How many functional requirements are covered in the operating scenarios? [ISO:5.4.2.g]
- What is the degree of confidence that the assumptions made on the environment are correct and sound? [ISO:5.4.2.c]
- Are all potential consequences of an item described? [ISO:5.4.1.f]

These concerns are based on the performance of the process and can be combined into:

IN 3: How well does the work comply with the, in the standard, defined requirements on the process?

Initiation of the safety lifecycle The objective of initiation of the safety lifecycle [ISO:6] is to determine if an item is a new development or a modification of an existing one. When an item is classified as a modification, the *Item Definition* from the previous step is reviewed and an impact analysis needs to be carried out. Otherwise, this step will be skipped. Therefore, the concerns in this phase are based on an assumption that an item is modified from an existing, fully specified and developed, item.

Some requirements in this step are identifying missing or incomplete work [ISO:6.4.2.9] which means existing work can be reused. The concern of the business manager and program manager is the amount of work that needs to be modified. Thus the following Information Need is derived:

IN 4: How much rework is required for redefining an item in terms of number of documents?

For the program manager, other concerns are focused on the item or subsystems of an item that require modification [ISO:6.4.2.{1-4, 8-9}]. More specifically the program manager has concerns for the required impact analysis. These concerns are:

- Are all modifications discovered and described?
- Are all implications of the modification known?
- Which work products are affected by the modifications and are they properly updated?

Besides, other concerns can be found related to the safety plan in ISO 26262 Part 2. The requirements on the safety plan in Part 3 [ISO:6.4.2.5-7] are focused on tailoring the safety plan and its activities according to the results from impact analysis. Hence a valid concern is:

IN 5: Are all results from the impact analysis incorporated in the safety plan and its activities?

Hazard analysis and risk assessment In this phase [ISO:7], two work products are created: *Hazard Analysis and Risk Assessment Report*, and *Safety Goals*. The objective of this clause is to avoid unreasonable risk.

For the program manager, the completeness of the hazard identified [ISO:7.4.2.1, ISO:7.4.2.2] is concerned. Thus the following Information Need is identified:

IN 6: Are the identified hazards complete?

When a hazard has a high risk of compromising the integrity of safety, it is classified as such. This is done using Automotive Safety Integrity Level (ASIL) [ISO:7.4.3-4]. The ASILs are based on three constraints: severity [ISO:7.4.3.1-3], probability of exposure [ISO:7.4.3.1-3] and controllability [ISO:7.4.3.1-3]. The classification consist of four levels, ASIL A, ASIL B, ASIL C and ASIL D. ASIL D indicates the highest integrity requirements on the product and ASIL A the lowest. The costs to address an ASIL D hazard is higher than an ASIL A hazard. Therefore, for a business manager, they are not only concerned with the number of hazards found, but also the number of hazards classified in each ASIL.

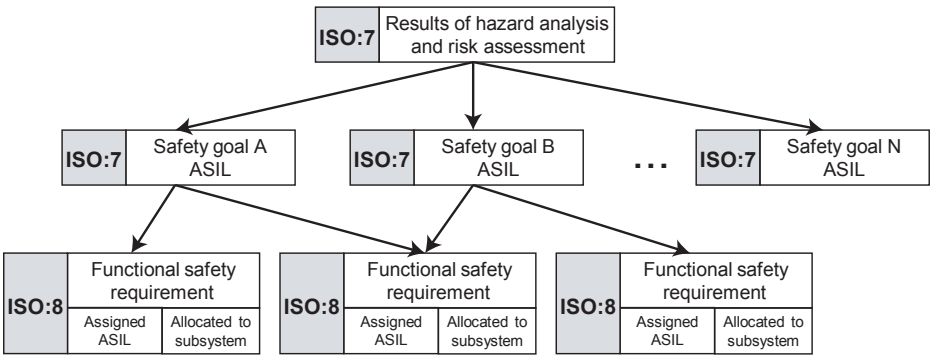


Figure 8.5: Hierarchy of safety goals and functional safety requirements based on [12, Part3, Figure 2].

IN 7: How many hazards are identified?
IN 8: How many hazardous events are identified for each ASIL?

Besides, the program manager has a responsibility to make sure that the classification has been done correctly, which is crucial for future development.

IN 9: Are hazards classified with a correct ASIL?

The last step in this phase is to create safety goals for the identified hazards [ISO:7.4.4.3]. The safety goals are assigned to an ASIL [ISO:7.4.4.3-4] according to the corresponding hazards. The Information Needs addressing concerns for this step are:

IN 10: Are all hazards transformed to safety goals?
IN 11: Are all safety goals classified with a correct ASIL?
IN 12: How many safety goals are identified for each ASIL?

Functional safety concept The functional safety concept [ISO:8] is a continuation of the previous clause. In this clause, safety goals are transferred to functional safety requirements. These functional safety requirements are allocated to a subsystem or component of the item being developed, or to external measures. The relation between safety goals and functional safety requirements is shown in Figure 8.5.

As mentioned before, the business manager is concerned with the amount of work delivered. Therefore, the following Information Need is added:

IN 13: How many functional safety requirements are derived?

For the functional safety requirements [ISO:8.4.2], the program manager is concerned with the number of safety goals covered by the functional safety requirements [ISO:8.4.2.1-2], and the level of detail of these functional requirements [ISO:8.4.2.3]. For each functional safety requirement, an ASIL will be assigned [ISO:8.4.3], this derives another Information Need related with ASIL. Therefore, there are three Information Needs added:

IN 14: Are all safety goals covered by the functional safety requirements?
IN 15: Are all relevant details discussed for the functional safety requirements?
IN 16: Does the associated ASIL of the functional safety requirements comply with the safety goals addressed?

8.5.2.2 Metrics Design

In this section, we discuss the metric design. The results of the metric design are shown in Figure 8.1. We use some examples to illustrate our approach. The changes of the ICM table proposed in this phase will be discussed in Section 8.5.2.3.

Categorization To design metrics for the Information Need, we first assign existing Information Categories in the ICM table to each Information Need. In this case, a Category will be selected or proposed if it covers the main topic of the concern. For example, for *IN 1* the main topic is resource usage and the costs connected to the resources. For this Information Need we selected the existing Information Category *Resource and Cost*. A number of Information Needs focus on documentation, hence we add the Category *Documentation* to the available Information Categories.

Measurable Concepts The next step is to identify the Measurable Concept. It could be an existing Measurable Concept in the ICM table or a new proposed one. For the *Documentation* category, the Measurable Concept is used to identify the document on which the metric is based. The concepts in the Category *Documentation* also represent the name of the document. An example of this is *IN 7*. For this Information Need, the number of hazards is required. This is documented in the *Hazard Analysis and Risk Assessment* document, which is also the name of the Measurable Concept.

Prospective Measures & Base Measures Prospective Measures and Base Measures are the basis of the final metric implementation. They provide information on the implementation and the sources of the measures. Similar to other steps, the ICM table is checked to find the relevant Prospective Measures and Base Measures. If necessary, new Prospective Measures and corresponding Base Measures can be added.

8.5.2.3 Modifications to the ICM table

In Table 8.1 the modified ICM table is presented. The changes are illustrated with an asterisk (*). In this section, we discuss all the modifications.

Information Categories We added a new Information Category *Documentation*. Information Needs classified within this Category have some concerns regarding documentation of a project. Examples of this could be the number of defined items, or the quality of documents in general. In our case study, the concerns of this Category are mainly about the number of defined items. The quality of the documents can be derived from audits of the process.

Measurable Concepts A new Measurable Concept, *Process Completeness*, in the Information Category *Process Performance* is introduced. This concept focuses on metrics which measure the completeness of the process. These metrics are often based on a relation table between two work products of the process.

For the *Documentation* category, four new Measurable Concepts are added: *Item Definition*, *Impact Analysis*, *Hazard Analysis and Risk Assessment*, and *Functional Safety Concept*. Each of these concepts measures the content of a given document. These metrics focus on the amount of work for creating or modifying the document or the information in the document.

Table 8.1: Adapted ICM Table based on [16], changes are marked with an asterisk (*).

IN	Information Need / Question Addressed	Information Category	Measurable Concept	Prospective Measure	Sample Base Measure
1	What are the staff costs of the entire process?	Resources and Cost	Financial Performance	- Earned Value Cost and Schedule Variance	- Actual Cost of Work Performed (ACWP)
2	What are the overhead costs for developing an item?	Resources and Cost	Facilities and Support Resources	- Cost of Extra Resources*	- Licensing costs* - Certification costs* - Legal costs*
3	How well does the work comply with the standard requirements on the process?	Process Performance	Process Compliance	- Process Audit Findings Distribution - Requirements Compliance Status*	- Number of audit findings regarding process - Requirements met by scenarios*
4	How much rework is required for redefining an item in terms of number of documents?	Documentation*	Impact Analysis*	- Rework Documents*	- Number of documents requiring rework*
5	Are all results from the impact analysis incorporated in the safety plan and its activities?	Process Performance	Process Completeness*	- Impacts to Safety Plan*	- Impacts met by the safety plan and its activities*
6	Are the identified hazards complete?	Process Performance	Process Compliance	- Process Audit Findings Distribution	- Number of audit findings regarding hazards*
7	How many hazards are identified?	Documentation*	Hazard Analysis and Risk Assessment*	- Number of Hazards*	- Number of hazards*
8	How many hazards are identified for each ASIL?	Process Performance	Process Completeness*	- Hazards to ASIL*	- Number of hazards per ASIL*
9	Are hazards classified with a correct ASIL?	Process Performance	Process Compliance	- Process Audit Findings Distribution	- Audit findings regarding ASIL*
10	Are all hazards transformed to safety goals?	Process Performance	Process Completeness*	- Hazard to Safety Goal*	- Number of hazards without a safety goal*
11	Are all safety goals classified with a correct ASIL?	Process Performance	Process Compliance	- Process Audit Findings Distribution	- Audits on the relation between the ASIL of hazards and the derived safety goals*
12	How many safety goals are identified for each ASIL?	Process Performance	Process Completeness*	- Safety goal to ASIL*	- Number of safety goals per ASIL*
13	How many functional safety requirements are derived?	Documentation*	Functional Safety Concepts*	- Number of Function Safety Requirments*	- Number of functional safety requirements*
14	Are all safety goals covered by the functional safety requirements?	Process Performance	Process Completeness*	- Safety Goal to Functional Requirement*	- Number of safety goals not covered by requirements*
15	Are all relevant details discussed for the functional safety requirements?	Process Performance	Process Compliance	- Process Audit Findings Distribution	- Audits on the details captured in functional requirements*
16	Does the associated ASIL of the functional safety requirments comply with the safety goals addressed?	Process Performance	Process Compliance	- Process Audit Findings Distribution	- Audits on the relation between the ASIL of the derived functional safety requirements and safety goals*

Prospective Measures & Base Measures In our case study, a number of new Prospective Measures and Base Measures are introduced. Those new Prospective Measures and their Base Measures can be grouped according to their characteristics.

The first group is based on relation mappings between different phases in the development process. For example, for the Information Need *IN 3*, a new Prospective Measure *Requirements Compliance Status* is added. This Prospective Measure indicates a mapping from requirements to operating scenarios. Then a new Base Measure named *Requirements met by scenarios* is suggested to capture this metric. Similar new Prospective Measures and Base Measures can be found for *IN 5*, *IN 8*, *IN 10*, *IN 12*, and *IN 14*.

Another group can be found for *IN 4*, *IN 7*, and *IN 13*. The Prospective Measures of these Information Needs all refer to a document, the name of which is provided by the Measurable Concept. The number of documents that require rework is measured for *IN 4*. This is done using *Impact Analysis* document. The number of hazards, the concern of *IN 7*, is measured based on the *Hazard Analysis and Risk Assessment*. For *IN 13* the Information Need is satisfied with the information available in the *Functional Safety Concepts* document.

Moreover, there is another new Prospective Measure *Cost of Extra Resources* added. This measure is used to measure the costs of extra resources. For *IN 2* we mainly focus on the legal costs of certain requirements, which can be measured by the metrics *Licensing costs*, *Certification costs*, and *Legal costs*. Finally, a few sample Base Measures for the Prospective Measure *Process Audit Findings Distribution* are proposed. Each of these Base Measures explains different Information Needs to be acquired from process audits. These additions can be found for Information Needs *IN 6*, *IN 9*, *IN 11*, *IN 15*, and *IN 16*.

8.5.3 Available Data: Common Certification Framework

As mentioned before, the common certification framework is designed and built based on a language called CCL. Therefore all the data is stored in a CCL-based database. That is to say, the data can be measured only if it can be described by CCL.

The CCL is a combination of number of different metamodels [26], e.g. Mapping, AssuranceAsset, Baseline, Process, and Argumentation metamodel. Given these metamodels, metrics can be derived, for instance, from the Process metamodel the metrics: number of (not started, ongoing, or finished) Activities can be derived. From the Argumentation metamodel metrics related to safety claims can be derived, such as number of (uninstantiated or undeveloped) safety claims. Finally, based on all the CCL metamodels, we designed 89 metrics via brainstorming.

8.5.4 Relevant Metrics

After combining the aforementioned 89 metrics with industrial interests and standard requirements, 76 metrics are finally selected. The 76 metrics are the intersection of *metrics based on available data* with *metrics from industrial partners*, unified with the intersection of *metrics based on available data* with *metrics from safety standard*. The details of those 76 metrics are documented in the OPENCOSS Deliverable D7.5 [27]. Moreover the common certification framework [61] was extended to visualize the metrics.

Table 8.2 shows the overlap between metrics from ISO 26262 Part 3 (in Figure 8.1) with metrics from other sources (industrial interests and available data). We could see that even for a part of the safety standard, the overlap among these three sources exists. Fifteen out of nineteen metrics derived from this part of the safety standard can be

Table 8.2: The overlap between metrics from ISO 26262 Part 3 (See Table 8.1) with metrics from other sources: I represents *Industrial interests*, D represents *available Data*, and - represents *no sources found*

Metrics from safety standards	**Overlap**
Actual Cost of Work Performed (ACWP)	I
Licensing costs	I
Certification costs	I and D
Legal costs	I
Number of audit findings regarding process	I and D
Requirements met by scenarios	I
Number of documents requiring rework	-
Impacts met by the safety plan and its activities	I
Number of audit findings regarding hazards	I
Number of hazards	I
Number of hazards per ASIL	-
Audit findings regarding ASIL	-
Number of hazards without a safety goal	I
Audits on the relation between the ASIL of hazards and the derived safety goals	I
Number of safety goals per ASIL	D
Number of functional safety requirements	I and D
Number of safety goals not covered by requirements	I and D
Audits on the details captured in functional requirements	I and D
Audits on the relation between the ASIL of the derived functional safety requirements and safety goals	I

mapped to the metrics from the industrial partners. Six out of these nineteen metrics are supported by available data. Moreover, five of the metrics can be mapped to both the metrics from industrial partners and metrics based on available data.

8.6 Validation

Set-up Some metrics share scales based on similar characteristics of the data. Therefore, the 76 metrics are put into 16 groups and each of these groups is displayed in separate charts or tables. For example, there are six metrics related with Baseline Metamodel (a part of the CCL): number of requirements for each applicability and criticality level (for instance ASIL A, ASIL B, ASIL C, and ASIL D), number of *Base Requirements*, number of *Base Activities*, number of *Base Techniques*, number of *Base Roles*, and number of base artefacts. Except for the first metric, the other five metrics can share a scale. We can divide these metrics into two groups: Baseline Metrics I, containing the number of requirements for each applicability and criticality level, and Baseline Metric II for the other five metrics.

The number of metrics implemented in each group is shown in Table 8.3. Figure 8.6 shows the graphical user interface of these metrics. We can see that the *Time Efficiency* metric group (the button with highlight background) is implemented as a histogram of all the activities. The platform users can monitor the status of all the activities via

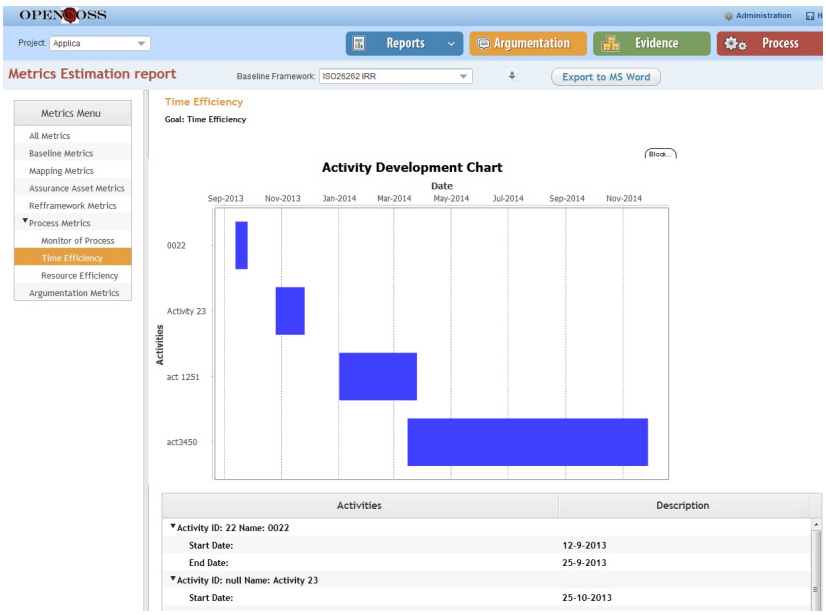


Figure 8.6: The graphical user interface of the metrics in the OPENCOSS platform.

Table 8.3: The number of metrics implemented in each metrics group.

Metrics Group	Number of Metrics
Equivalence Map Metrics	15
Baseline Metrics I	1
Baseline Metrics II	5
Mapping Metrics	15
Assurance Asset Metrics I (Evidence)	5
Assurance Asset Metrics I (Process)	5
Assurance Asset Metrics II (Evidence)	1
Assurance Asset Metrics II (Process)	1
Reframework Metrics	10
Monitor of Process Metrics I	2
Monitor of Process Metrics II	3
Monitor of Process Metrics III	3
Time Efficiency Metrics	1
Resource Efficiency Metrics	1
Argumentation Metrics I	5
Argumentation Metrics II	3

the histogram, and the details of these activities are also given in the table below the histogram.

Survey For the validation of the collected metrics, a survey has been carried out by 24 experts from the 13 project partners. These experts consist of safety engineers, safety managers, safety assessors, and researchers. To minimize the time spent on the survey

and visualize the relations between the metrics, the 76 metrics are evaluated in groups (as shown in Table 8.3). During the survey, those experts were asked to give an evaluation score to the metrics groups. The evaluation scores are designed as following:

- 5: Very useful. The metrics in this chart or table are very useful. They are mandatory to be implemented in the framework.
- 4: Useful. The metrics in this chart or table are useful in most situations. They are highly recommended to be implemented in the framework.
- 3: Moderately useful. The metrics in this chart or table are useful in some situations. They can be implemented in the framework.
- 2: Slightly useful. The metrics in this chart or table are rarely used. They can either be implemented in the framework or not.
- 1: Useless. The metrics in this chart or table are useless. They are highly recommended not to be implemented in the framework.
- 0: Completely Useless. The metrics in the chart or table are very useless. They do not need to be implemented in the framework.

After collecting the questionnaires, we got the overall evaluation results, which are shown in Figure 8.7. For example, for the *Time Efficiency* metric, the average evaluation score is 3.68, which means the experts think this metric is useful based on their knowledge

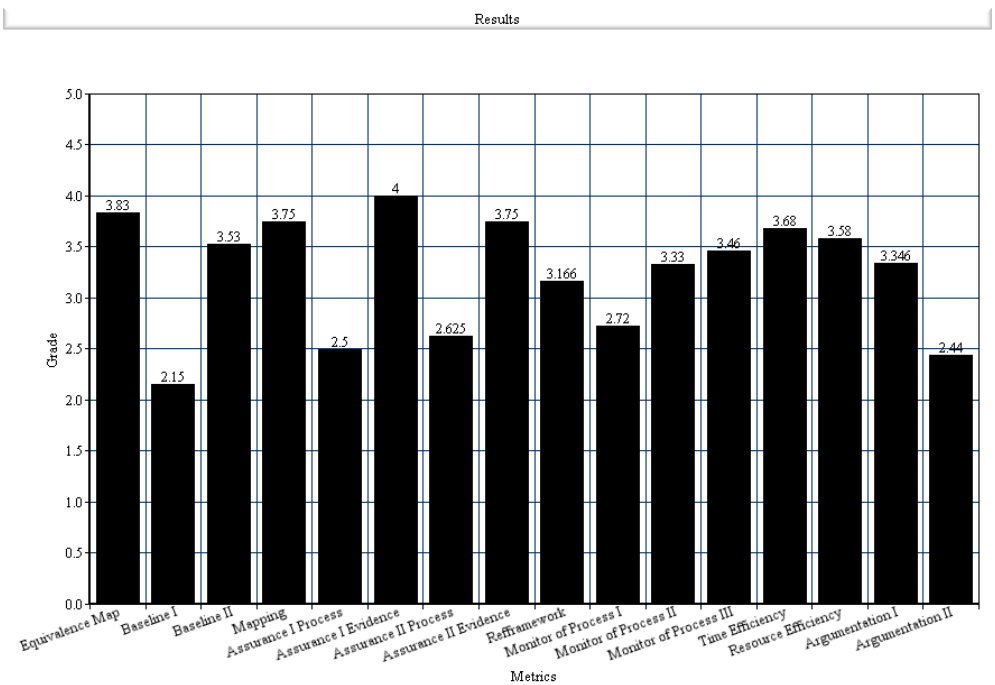


Figure 8.7: The validation results on the final metrics.

and experience. Besides, we can see that 11 groups got an average evaluation score above 3 (Moderately useful), and the evaluation scores of all groups are above 2 (Slightly useful).

Discussion From the result of the validation, we could see that most of these metrics are useful for the project partners. We think the reason for this is that there is a big overlap between metrics from industrial partners and metrics based on database. About 42% metrics from industrial partners are also included in the database. When industrial partners were asked to evaluate the metrics, they tend to give the score based on their interests, experience, and knowledge. Therefore, if the metrics are also included in the metrics from industrial partners, they can get a high score. For example, for the group Assurance I Evidence, it got a average evaluation score 4, which is the highest score among all groups. The metrics in this group are number of events (creation, modification, evaluation, approval, revocation) for evidence per week. These metrics can also be mapped to the metrics from industrial partners. For the group Baseline I, it got the lowest average evaluation score 2.15. The metric in this group is number of requirements for each applicability and criticality levels (for instance ASIL A, ASIL B, ASIL C, and ASIL D). This metric can also be derived from safety standards, but it is not included in the metrics from industrial partners. This means when deriving metrics from industrial interests, industrial partners tend to give generic metrics used in their domain instead of specific metrics from safety standards.

Effort When applying our approach, human work is involved in some activities, such as design of a survey, analysis the survey results, interpretation of safety standards, obtaining relevant metrics etc. Because a high number of metrics obtained, the method for metrics evaluation needs to be carefully selected and the results of the evaluation should be collected and discussed. Consequently, this may also involve a considerable human effort.

Threats to Validity There are some threats to validity related to the approach present in this chapter. First, we used three different sources for designing metrics for safety assessment process, the results can be affected if any of these sources changes. For instance, for the industrial interests, if different industrial partners are involved in the survey for obtaining metrics, a different set of metrics might be selected. Consequently, the final relevant metrics might be affected. However, the process of our methodology is not affected by these changes. Second, metrics obtained from different sources might be the same but with different names, for example number of safety goals and number of safety claims. The goal of both metrics is to measure the number of goals in a safety argumentation. If these two metrics are treated as different metrics, we might either introduce duplication to the results by selecting both of them as the relevant metrics or miss the overlap between them and not selecting them as relevant metrics.. Therefore, we use human interpretation to address this issue. Our methodology allows flexibility with respect to the collection and processing of metrics by human beings.

8.7 Related Work

As discussed in the Section 8.1, some research can be found on designing metrics. GQM is a common approach for metric design [41]. Based on GQM, Practical Software and Systems Measurement (PSM) has been developed to deal with modern software, management challenges and coordination of safety processes. The PSM framework also has been

applied to the safety domain [109]. Moreover, there are a number of standards proposed to support the methodologies of software measurement, such as ISO/IEC 15539 [81] and IEEE 1061-1998 [80]. In this chapter, our methods are mainly based on these existing approaches and standards.

Model-driven software engineering (MDSE) is widely used for the development of software systems. The Object Management Group (OMG) has proposed a number of standards to support model-driven architecture, such as MOF (Meta Object Facility) [117], XML Metadata Interchange (XMI) [115] and Unified Modeling Language (UML) [143]. These architectures allow developers to build systems using metamodels and models. Therefore, to measure these metamodels and models, model-based approaches has been advocated to designing metrics. Software Metrics Metamodel (SMM) [28] is a metamodel published by the OMG to describe standard software metrics. An SMM model describes how metrics can be extracted from a model. Based on the SMM, model-driven measurement approach has been proposed [107]. This approach allows modelers to automatically add measurement capabilities to a domain-specific modeling language. Our approach is not based on the SMM, but it could be an interesting extension of our approach.

There is a huge amount of publications describing metrics for software and models both for product and process. These metrics are used for measuring complexity, quality, development effort, maintenance effort, etc. There are considerable less existing published work that deal with metrics and safety assessment. This is because safety assessment is a relatively young domain. Other related work can be found on safety metrics design. Safety metrics are used to manage safety culture (or performance of safety activities) in an organization [83]. The safety culture management can include program development, benchmarking, auditing, measuring performance, etc. Glendon and Stanton have proposed two approaches for measuring safety culture [73]. Moreover, an assessment tree method is proposed for quantitative estimation of safety culture level in an organization [148]. However, in this chapter we only focus on metrics of safety assessment process, which is strongly related with safety standards.

8.8 Conclusion

Safety, assured by standards, is a critical issue for the transportation domain. The goal of safety assessment is to check that the final product/system complies with specific standards for safety. In this study we proposed three questions (*What do we want to measure? What should we measure? What can we measure?*) for designing metrics for safety assessment process. To answer those questions, three sources are identified: industrial interests, safety standards, and available data. Different methods can be used for obtaining metrics from these sources. A survey can be used to define metrics according to industrial interests. A method can be proposed for adapting the PSM framework to include safety assessment metrics obtained from the relevant safety standards. Finally, other metrics can be derived from the available data for a given project or in some database.

To demonstrate our approach, a case study has been carried out in the context of OPENCOSS. The part of safety standard metrics has been done on ISO 26262 Part 3 [12]. Fifteen out of nineteen metrics derived from this part of the safety standard can be mapped to the metrics from the industrial partners. Six out of these nineteen metrics are supported by available data. Moreover, five of the metrics can be mapped to both the metrics from industrial partners and metrics based on available data. After obtaining

the relevant metrics, the OPENCOSS framework has been extended to visualize these metrics. Finally, a validation on those metrics has been done with 24 experts from 13 project partners. The results of the validation show that most of the metrics are useful for industry.

Chapter 9

Conclusions

This chapter concludes this thesis by discussing the main contributions and directions for future research. For each of the research questions stated in Chapter 1, we provide the main results and conclusions. Additional details are available in the chapters that cover the research questions.

9.1 Contributions

The main research question covered in this thesis is formulated as follows.

> **RQ:** *Can model-driven techniques support the current safety assurance processes?*

This question is divided into six more specific research questions, and each of these questions is addressed in the chapters of this thesis.

The first of these questions deals with efficient extracting models from safety standards and is formulated as follows.

> **RQ$_1$:** *How can models for safety standards be created efficiently?*

To address this question, we presented a model-based approach to enabling safety assurance reuse through objective and cost-efficient modeling of the relevant standards, as described in Chapter 2. The Snowball methodology provides rules for extracting the conceptual model from a safety standard, thereby reducing the amount of manual work involved. As a case study, we investigated ISO 26262 part 3. Over 90% of the concepts in the industrial models are covered by our conceptual model. A better result could be obtained if the domain experts were involved in all the steps of the Snowball approach, but it will also be more costly. Besides, the availability of a rule based approach can reduce up to 80% of the amount of time required to validate the final model by the domain experts. The process in the standard is modeled with the OMG Software & Systems Process Engineering Metamodel (SPEM). Although the approach currently operates only

at a very high level, it provides a basis for describing a process model in the context of the safety standards.

> **RQ_2:** *How to support GMM evolution and refinement for different safety domains?*

We addressed this question in Chapter 3 and 4 through a metamodel transformation approach to facilitate safety assurance. To support evolution and modification of the generic metamodel, a refinement process for a generic metamodel according to a system supplier's input is presented in Chapter 3. A graphical editor can be generated based on the resulting specific metamodel. Therefore, our approach not only enables the user, such as the system supplier, to reuse the existing certification data by means of a conceptual mapping, which is supported by the GMM, but it also respects their current way of working by means of specific metamodels support.

Besides, a metamodel refinement language is defined. With its help, the traceability information from the GMM to SMMs can be maintained, as described in Chapter 4. Refinement specifications support the documentation of the traces between generic and more specific metamodels and vice versa. They can be used to automatically determine the similarity between concepts in different domains. The study of comparative mapping support between conceptual models or metamodels in safety-critical domains is a promising approach to improve the understanding between different domains or companies at the conceptual level and, consequently, the reuse of safety assurance data at the model level. The traceability information is stored in metamodels themselves for easy maintenance and analysis. Meanwhile for each refinement of metamodels, a mapping specification is generated along with the target metamodels. A comparison between different conceptual models or metamodels can be obtained by analyzing the related mapping specifications. Moreover, conceptual models can also be used for constructing safety cases. In this case, the mappings found between conceptual models can support safety case reuse.

> **RQ_3:** *Can we reduce the ambiguities in the safety case by using a controlled language?*

To address this question, in Chapter 5 we investigated using controlled language to express safety cases. A vocabulary-based methodology is presented for safety case development. There are four main contributions of our approach: Firstly, an SBVR editor is implemented. This editor supports the development of SBVR vocabularies and rules, and provides graphical editing and checking of a vocabulary. Secondly, a model transformation from an EMF model to an SBVR model is proposed. A vocabulary can be automatically generated from a conceptual model, which facilities the vocabulary creation. Thirdly, by using SBVR vocabulary, it enables the explicit connection between conceptual models and safety cases to ensure that certification data is built properly and can be reused efficiently. Finally, by utilizing SBVR, the content of safety case elements is well-structured and well-controlled. It can reduce mistakes and misunderstanding between the different roles involved in producing, assessing, and using the safety case. Thus, our method supports improving the clarity and correctness of safety cases, and increasing the confidence in the claimed safety assurance.

> **RQ_4:** *How to collect the required evidence for safety claims efficiently?*

In Chapter 6, we presented a model-based approach to support evidence collection in safety certification, which is a basis for semi-automatic or automatic evidence collection in

the future. Our approach supports the evidence collection through four phases: evidence requirement analysis, traceability management, evidence collection preparation, and evidence collection. There are a number of benefits of our systematic approach. Firstly, the safety case conceptual model provides a model-based representation of evidence requirements, which reduces the potential ambiguity and inconsistency in textual safety claims. As a result, the risk of recollecting evidence, due to misunderstanding of evidence requirements, is mitigated. Besides, in the traceability management phase, the traceability between the safety claims and the collected evidence helps suppliers find the potential shared evidence items between objectives. Thus, the re-usability of evidence can be improved and the cost of repeatedly collecting evidence can be reduced. Finally, the traceability information could also be used to link the collected evidence items to the corresponding objectives. Another contribution of our approach is to detect evidence artifacts missing in an early phase so that the confidence of the collected evidence can be improved.

> **RQ$_5$:** *How to manage functional safety according to ISO 26262 for an existing safety-critical system?*

To address this question, we introduced a methodology to improve functional safety for an existing automotive system, according to ISO 26262, as described in Chapter 7. It includes five main phases: Review and Plan phase, Conceptual phase, Design phase, Implementation phase, and Verification and Validation phase. Our main goals is to apply ISO 26262 to an existing system and to maximize the reuse of the existing project data. An industrial case study from TNO has been used to illustrate our approach in a detailed way. The internal validation results show that for the original architecture design of the system, only four out of these ten test cases passed. For the new design, however, eight of these ten test cases passed. Based on this, we conclude that our approach can facilitate the improvement of the system architecture design. Finally, an external validation by domain experts has been performed for our case study.

> **RQ$_6$:** *How to design metrics for safety assessment process?*

Safety is a critical issue for the transportation domain which is assured with standards. Safety assessment needs to be performed to ensure the safety of systems. However, the process of safety assessment is usually costly and time-consuming. In Chapter 8 we have proposed a method for designing metrics for safety assessment. By applying the method, three questions have been answered: *What do we want to measure? What should we measure? What can we measure?* Metrics for safety assessment can be derived from three sources. Different methods for designing metrics can be used for each source. For the demonstration of our approach, a case study in the context of the European project OPENCOSS is carried out. Finally, a validation of the obtained relevant metrics has been done by means of a survey amongst 24 experts from 13 project partners. The validation results show that industrial partners are satisfied with the final metrics.

Summary In conclusion, this thesis addressed the main research question from three aspects: standard-based approach, argument-based approach, and safety assurance and certification process.

For the standard-based approach, we have presented a rule-based solution for modeling and analyzing safety standards (see Chapter 2). Besides, we also carried out research on

the evolution and refinement of the CCL (see Chapter 3 and Chapter 4). For argument-based approach, we have introduced a controlled-language to construct safety cases to reduce the ambiguities (see Chapter 5). Moreover, the evidence collection for safety cases has also been studied (see Chapter 6). For safety assurance and certification process, we have applied the ISO 26262 to facilitate functional safety management of existing systems (see Chapter 7). Besides, we have proposed a process for designing metrics for safety assessment (see Chapter 8).

9.2 Future Work

In this thesis we present a number of model-based approaches to facilitate safety assurance and certification. This section presents some interesting directions for future work.

Standard-based Approach. In this thesis, two approaches are described to get conceptual models for safety standards. One is manually extraction as described in Chapter 2, one is automatic generation from a generic metamodel as described in Chapter 3.

For the manual extraction, a rule-based approach is proposed. The first direction of the future work is to extend this approach for different safety standards, such as DO 178C. Another future direction is to define a domain specific meta-model for the processes found in safety standards, since Software & Systems Process Engineering Metamodel (SPEM) is too general to describe the process model of the safety standards at the lower level. Adedjouma has described an extension of SPEM for ISO 26262 [31]. However, for different standards, different extensions might be required. The last future direction is to use existing Natural Language Processing (NLP) techniques [50] and ontology learning methodologies to support the Snowball approach and further reduce the amount of manual work involved in the implementation of the approach.

For the automatic generation, a metamodel refinement language is developed to support metamodel transformation in Chapter 3. In the future, the concrete syntax of the MMRL can be improved to make it more user-friendly for the user, such as safety mangers and system suppliers. In some domains (such as automotive and avionics) similar systems have been developed according to different safety standards. This fact could be exploited to reuse safety assurance arguments from one system to the other, which necessarily involves a comparative study of the two respective standards. A high-level comparison between two safety standards (ISO 26262 and DO 178B) is outlined by Gerlach et al. [71]; but the challenge here is to compare the standards at the lower level to support safety assurance reuse. This could be achieved through conceptual mapping and process mapping. To address this, a study on comparative mappings of safety standards via generic metamodel is carried out in Chapter 4. In the future, the applications of comparative mappings can be studied and implemented, for example, how to use comparative mappings between metamodels to facilitate safety case or evidence reuse. In addition, detecting comparative mappings in the model level automatically or semi-automatically is also promising.

Argument-based Approach. To reduce ambiguities in safety cases, in Chapter 5, SBVR controlled English is introduced to express the content of safety cases. As

future work, this approach can be extended by introducing SBVR for safety case pattern development. Moreover, more large scale applications from industrial partners can be used as case studies. Based on the research in Chapter 5, Chapter 6 discussed how to extract conceptual models from SBVR-based safety claims to facilitate safety evidence collection. However, the rules for extracting conceptual models from safety claims can be improved, and the methodology can be extended so that it can support safety evidence reuse in different domains. Another direction could be the tool support for semi-automatic or automatic evidence collection.

Safety Assurance and Certification Process. We have carried out some study on the overall management of safety assurance and certification process. In Chapter 7, we discussed how to use ISO 26262 for functional safety improvement of the existing safety-critical systems. An industrial case study in the context of automated driving is given. In the future, further exploration of the effectiveness and extensibility of our approach can be carried out by applying it to more industrial case studies. Moreover, the generalization of our approach to other safety-critical domains, like the railway domain or the avionic domain, can be investigated.

To assess both efficiency and effectiveness of the certification process, metrics are required. In Chapter 8, based on three sources of information (industrial interests, safety standards, available data), a number of metrics are developed for safety assurance and certification process. However, the current metric design and implementation are all done manually and lack transparency. Hence, another research direction could be to use model-driven techniques to facilitate metric design and implementation to reduce manual cost and make this process transparent.

Bibliography

[1] dSPACE Website. www.dspace.com, accessed Nov 2015.

[2] Eclipse Modeling Framework. https://eclipse.org/modeling/emf/, accessed Nov 2015.

[3] Eclipse Process Framework Project. http://www.eclipse.org/epf/.

[4] EuGENia. http://www.eclipse.org/epsilon/doc/eugenia/, accessed Nov 2015.

[5] iFast. http://www.artemis-ifest.eu/home, accessed Nov 18, 2014.

[6] Tube Lines Contractual Safety Case.

[7] Xtext. https://eclipse.org/Xtext/, accessed Nov 2015.

[8] DO 178B: "Software Considerations in Airborne Systems and Equipment Certification", 1992.

[9] EUR Whole Airspace ATM System Safety Case, 2001. http://dependability.cs.virginia.edu/research/safetycases/EUR_WholeAirspace.pdf.

[10] BBC NEWS: Toyota car recall may cost $2bn, 2010. http://news.bbc.co.uk/2/hi/business/8493414.stm, accessed August 2015.

[11] IEC 61508: "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems", 2010.

[12] ISO 26262: "Road Vehicles – Functional Safety", 2011.

[13] Official Report to the State Council on the Cause of the Accident and Recommendations Regarding Accident Prevention and Corrective Measures (Chinese), 2011. http://news.sina.com.cn/c/2011-12-28/201223711187.shtml, accessed August-2015.

[14] RTCA DO-178C: Software Consideration in Airborne Systems and Equipment Certification, Dec 2011.

[15] Automated Defect Prevention for Embedded Software Quality, 2012. White Paper by VDC Research.

[16] Information Category-Measurable Concept-Prospective Measures (ICM) Table v7.0a final, 2012.

[17] MIL-STD-822E, 2012. http://www.system-safety.org/Documents/MIL-STD-882E.pdf.

[18] OPENCOSS: Deliverable D2.2 - High-level requirements (report), 2012. http://www.opencoss-project.eu/node/7.

[19] OPENCOSS: Deliverable D4.1 - Baseline for the common certification language, 2012. http://www.opencoss-project.eu/node/7.

[20] OPENCOSS: Deliverable D4.2 - Detailed requirements for the common certification language, 2012. http://www.opencoss-project.eu/node/7.

[21] Meta Modeling Approach to Safety Standard for Consumer Devices, 2013. http://www.omg.org/news/meetings/tc/agendas/ut/SysA_Slides/taguchi.pdf.

[22] OPENCOSS: Deliverable D4.3 - Intermediate common certification language conceptual model, 2013. http://www.opencoss-project.eu/node/7.

[23] OPENCOSS: Deliverable D7.4 - Specification of the transparent certification service infrastructure, 2013. http://www.opencoss-project.eu/node/7, accessed August-2015.

[24] Safety Case Repository, 2013. http://dependability.cs.virginia.edu/info/SafetyCases:Repository.

[25] Structured Assurance Case Metaodel (SACM), version 1.0, 2013.

[26] OPENCOSS: Deliverable D4.4 - Common Certification Language: Conceptual Model, 2014. http://www.opencoss-project.eu/node/7.

[27] OPENCOSS: Deliverable D7.5 - Implementation of the process-specific service infrastructure, 2014. http://www.opencoss-project.eu/node/7, accessed August-2015.

[28] Software Metrics Meta-Model (SMM) Version 1.1, 2015. URL: http://www.omg.org/spec/SMM/1.1/.

[29] ASCOS: Aviation Safety and Certification of new Operations and Systems, Accessed July 2015. http://www.ascos-project.eu.

[30] SafeCer: Safety Certification of software-intensive systems with reusable components, Accessed July 2015. http://www.safecer.eu/.

[31] Morayo Adedjouma. *Requirements Engineering Process According to Automotive Standards in a Model-Driven Framework*. PhD thesis, Université Paris Sud - Paris XI, 2012.

[32] Nico Adler, Stefan Otten, Markus Mohrhard, and Klaus D. Muller-Glaser. Rapid Safety Evaluation of Hardware Architectural Designs Compliant with ISO 26262. In *Rapid System Prototyping (RSP), 2013 International Symposium on*, pages 66–72. IEEE, 2013.

[33] Hina Afreen, Imran S. Bajwa, and Behzad Bordbar. SBVR2UML: A Challenging Transformation. In *Frontiers of Information Technology (FIT), 2011*, pages 33–38. IEEE, 2011.

[34] Marcel F. Amstel, M. G. J van den Brand, and Alexander Serebrenik. Traceability Visualization in Model Transformations with TraceVis. In *Theory and Practice of Model Transformations*, volume 7307, pages 152–159. 2012.

[35] Grigoris Antoniou and Frankvan Harmelen. Web Ontology Language: OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 91–110. Springer Berlin Heidelberg, 2009.

[36] Eric Armengaud, Quentin Bourrouilh, Gerhard Griessnig, Helmut Martin, and Peter Reichenpfader. Using the CESAR Safety Framework for Functional Safety Management in the Context of ISO 26262. Embedded Real Time Software and Systems, 2012.

[37] Colin Atkinson and Thomas Kühne. Model-Driven Development: a Metamodeling Foundation. volume 20, pages 36–41. 2003.

[38] Imran Sarwar Bajwa, Mark G. Lee, and Behzad Bordbar. SBVR Business Rules Generation from Natural Language Specification. In *AAAI 2011 Spring Symposium - AI for Business Agility*, pages 2–8, San Francisco, USA, 2011.

[39] Matthew R. Barry. CertWare: A Workbench for Safety Case Production and Analysis. pages 1–10. IEEE, 2011.

[40] Victor Basili, Jens Heidrich, Mikael Lindvall, Jürgen Münch, Myrna Regardie, Dieter Rombach, Carolyn Seaman, and Adam Trendowicz. GQM+ Strategies: A comprehensive methodology for aligning business strategies with software measurement. 2014.

[41] Victor Basili and David Weiss. A Methodology for Collecting Valid Software Engineering Data. volume SE-10, pages 728–738. Nov 1984.

[42] Peter Bishop and Robin Bloomfield. A Methdology for Safety Case Development. 1998.

[43] David Brothanek, Martin Jung, Verena Jung, Michael Krell, Reinhard Pfundt, Elke Salecker, Ingo Strumer, and Heiko Zatocil. Process for Functional Safety, Model- based Software Development for Electric Drivetrains according to ISO 26262. *dSPACE Magazine*, pages 12–19, 2013.

[44] Jordi Cabot, Raquel Pau, and Ruth Raventós. From UML/OCL to SBVR specifications: A challenging transformation. volume 35, pages 417–440. 2010.

[45] CENELEC. EN50126: Railway Applications–The Specification and Demonstration of Reliability. *Availability, Maintainability and Safety (RAMS)*, 1999.

[46] CENELEC. EN50129: Railway Application–Safety Related Electronic Systems for Signaling, 2000.

[47] CENELEC. EN50128: Railway Applications-Communication, Signaling and Processing Systems-Software for Railway Control and Protection Systems. 2011.

[48] Lina Ceponiene, Lina Nemuraite, and Gediminas Vedrickas. Semantic Business Rules in Service Oriented Development of Information Systems. In *15th International Conference on Information and Software Technologies, IT*, pages 404–416, 2009.

[49] Peter P. Chen, Bernhard Thalheim, and Leah Y. Wong. Future Directions of Conceptual Modeling. In *Conceptual Modeling*, volume 1565 of *Lecture Notes in Computer Science*, pages 287–301. Springer Berlin Heidelberg, 1999.

[50] Gobinda G. Chowdhury. Natural Language Processing. volume 37, pages 51–89. Wiley Subscription Services, Inc., A Wiley Company, 2003.

[51] Antonio Cicchetti, Davide Di Ruscio, Romina Eramo, and Alfonso Pierantonio. Meta-model Differences for Supporting Model Co-evolution. In *Proceedings of the 2nd Workshop on Model-Driven Software Evolution-MODSE*, pages 1–10, 2008.

[52] Trevor Cockram and Ben Lockwood. Electronic Safety Cases: Challenges and Opportunities. In *Current Issues in Safety-Critical Systems*, pages 151–162. Springer London, 2003.

[53] Mirko Conrad. Verification and Validation to ISO 26262: A Workflow to Facilate the Development of High-integrity Systems. In *Embedded REal Time Software and Systems (ERTS2)*, pages 1–8, 2012.

[54] JoseLuis de la Vara, Sunil Nair, Eric Verhulst, Janusz Studzizba, Piotr Pepek, Jerome Lambourg, and Mehrdad Sabetzadeh. Towards a Model-Based Evolutionary Chain of Evidence for Compliance with Safety Standards. In Frank Ortmeier and Peter Daniel, editors, *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, pages 64–78. Springer, Heidelberg, 2012.

[55] Maurizio de Tommas and Angelo Corallo. SBEAVER: A Tool for Modeling Business Vocabularies and Business Rules. pages 1083–1091. Springer, 2006.

[56] Ewen Denney, Ganesh Pai, and Josef Pohl. AdvoCATE: An Assurance Case Automation Toolset. In *SAFECOMP Workshops*, volume 7613, pages 8–21. Springer, 2012.

[57] Stephen Drabble. Safety Process Measurement - Are we there yet? In Chris Dale and Tom Anderson, editors, *Safety-Critical Systems: Problems, Process and Practice*, pages 195–207. Springer London, 2009. URL: `http://dx.doi.org/10.1007/978-1-84882-349-5_13`.

[58] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting Empirical Methods for Software Engineering Research. In Forrest Shull, Janice Singer, and DagI.K. Sjøberg, editors, *Guide to Advanced Empirical Software Engineering*, pages 285–311. Springer London, 2008.

[59] Christof Ebert. Implementing Functional Safety. volume 32, pages 84–89. Sept 2015.

[60] Khaled El Emam, Walcelio Melo, and Jean-Normand Drouin. *SPICE: The theory and practice of software process improvement and capability determination.* IEEE Computer Society Press, 1997.

[61] Huáscar Espinoza, Alejanra Ruiz, Mehrdad Sabetzadeh, and Paolo Panaroni. Challenges for an Open and Evolutionary Approach to Safety Assurance and Certification of Safety-Critical Systems. In *Software Certification (WoSoCER), 2011 First International Workshop on*, Hiroshima, Japan.

[62] EUROCONTROL. Safety Case Development Manual, 2006. `http://www.eurocontrol.int/sites/default/files/article/content/documents/nm/link2000/safety-case-development-manual-v2.2-ri-13nov06.pdf`.

[63] Davide Falessi, Mehrdad Sabetzadeh, Lionel Briand, Emanuele Turella, Thierry Coq, and RajwinderKaur Panesar-Walawege. Planning for Safety Evidence Collection: a Tool-Supported Approach Based on Modeling of Standards Compliance Information. 2011.

[64] Davide Falessi, Mehrdad Sabetzadeh, Lionel Briand, Emanuele Turella, Thierry Coq, and RajwinderKaur Panesar-Walawege. Planning for Safety Standards Compliance: A Model-Based Tool-Supported Approach. volume 29, pages 64–70. May 2012.

[65] Jean-Rémy Falleri, Marianne Huchard, Mathieu Lafourcade, and Clémentine Nebut. Metamodel Matching for Automatic Model Transformation Generation. In *Model Driven Engineering Languages and Systems*, pages 326–340. Springer, 2008.

[66] Arlene Fink. *The Survey Handbook*, volume 1. Sage Publications, 2003.

[67] Arlene Fink. *How to Conduct Surveys: A Step-by-Step Guide.* Sage Publications, 2012.

[68] Floyd J Fowler Jr. *Survey Research Methods.* Sage Publications, 2013.

[69] Bernd Freimut, Teade Punter, Stefan Biffl, and Marcus Ciolkowski. State-of-the-art in empirical studies. Report: ViSEK/007/E, Fraunhofer Inst. of Experimental Software Engineering, 2002.

[70] Kelly Garcés, Frédéric Jouault, Pierre Cointe, and Jean Bézivin. Managing model adaptation by precise detection of metamodel changes. In RichardF. Paige, Alan Hartman, and Arend Rensink, editors, *Model Driven Architecture - Foundations and Applications*, volume 5562 of *Lecture Notes in Computer Science*, pages 34–49. Springer Berlin Heidelberg, 2009.

[71] Matthias Gerlach, Robert Hilbrich, and Stephan Weißleder. Can Cars Fly? From Avionics to Automotive: Comparability of Domain Specifc Safety Standards. In *Proceedings of the Embedded World Conference*, pages 1–9, March 2011.

[72] Ralf Gitzel and Tobias Hildenbrand. A Taxonomy of Metamodel Hierarchies. 2005.

[73] Aleck Ian Glendon and Neville A Stanton. Perspectives on Safety Culture. volume 34, pages 193–214. Elsevier, 2000.

[74] PatrickJohn Graydon. Towards a Clearer Understanding of Context and Its Role in Assurance Argument Confidence. In *Computer Safety, Reliability, and Security*, volume 8666 of *Lecture Notes in Computer Science*, pages 139–154. Springer International Publishing, 2014.

[75] Robert M Groves, Floyd J Fowler Jr, Mick P Couper, James M Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey Methodology*, volume 561. John Wiley & Sons, 2011.

[76] Thomas R Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing? volume 43, pages 907–928. Elsevier, 1995.

[77] GSN Community Standard: Version 1; November 2011, (c) 2011 Origin Consulting (York) Limited, 2011. `http://www.goalstructuringnotation.info/documents/GSN_Standard.pdf/`.

[78] Ibrahim Habli and Tim Kelly. A Model-Driven Approach to Assuring Process Reliability. In *Proceedings of ISSRE 2008*, pages 7–16, Washington, DC, USA, 2008. IEEE Computer Society.

[79] Richard Hawkins, Tim Kelly, John Knight, and Patrick Graydon. A New Approach to Creating Clear Safety Arguments. In *Advances in systems safety*, pages 3–23. Springer, 2011.

[80] IEEE. IEEE Standard for a Software Quality Metrics Methodology. IEEE 1061:1998, Institute of Electrical and Electronics Engineers, 1998.

[81] ISO/IEC. Systems and software engineering -Measurement process. ISO/IEC 15939:2007, International Organization for Standardization /International Electrotechnical Commission, Geneva, Switzerland, 2007.

[82] Daniel Jackson, Martyn Thomas, and Lynette I. Millet. *Software for Dependable Systems: Sufficient Evidence?* The National Academies Press, Washington, D.C., 2007.

[83] Christopher A Janicak. *Safety Metrics: Tools and Techniques for Measuring Safety Performance*. Government Institutes, 2009.

[84] S. Manoj Kannan, Yanja Dajsuren, Yaping Luo, and Ion Barosa. An Approach for Functional Safety Improvement of an Existing Automotive System. In *Proceedings of International Workshop on Modelling in Automotive Software Engineering*, 2015.

[85] Kate Kelley, Belinda Clark, Vivienne Brown, and John Sitzia. Good Practice in the Conduct and Reporting of Survey Research. volume 15, pages 261–266. ISQHC, 2003.

[86] Tim Kelly. *Arguing Safety - A Systematic Approach to Managing Safety Cases*. PhD thesis, University Of York, 1998.

[87] Tim Kelly. *Arguing Safety: A Systematic Approach to Managing Safety Cases*. PhD thesis, University of York, 1999.

[88] Tim Kelly and Rob Weaver. The Goal Structuring Notation - A Safety Argument Notation. 2004.

[89] Barbara Kitchenham, Shari Lawrence Pfleeger, Lesley M Pickard, Peter W Jones, David C Hoaglin, Khaled El Emam, Jarrett Rosenberg, et al. Preliminary Guidelines for Empirical Research in Software Engineering. volume 28, pages 721–734. IEEE, 2002.

[90] Martin Krammer, Eric Armengaud, and Quentin Bourrouilh. Method Library Framework for Safety Standard Compliant Process Tailoring. In *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 302–305. IEEE, 2011.

[91] J. Langheim, B. Guegan, L. Maillet-Contoz, K. Maaziz, G. Zeppa, F. Phillipot, S. Boutin, I. Aboutaleb, and P. David. System Architecture, Tools and Modelling for Safety Critical Automotive Applications - The R&D Project SASHA. In *ERTS2 2010, Embedded Real Time Software & Systems*, pages 1–8, Toulouse, France, 2010.

[92] Robert Lewis. Safety Case Development as an Information Modelling Problem. In Chris Dale and Tom Anderson, editors, *Safety-Critical Systems: Problems, Process and Practice*, pages 183–193. Springer London, 2009.

[93] Huan Lin, Yaping Luo, Ji Wu, Chunchun Yuan, M. G. J van den Brand, and Luc Engelen. A Systemtic Approach for Safety Evidence Collection in the Safety-Critical Domain. In *8th Annual IEEE System Conference*, pages 194–199, 2015.

[94] Yaping Luo, Luc Engelen, and M. G. J van den Brand. Metamodel Comparison and Model Comparison for Safety Assurance. In *Computer Safety, Reliability, and Security*, volume 8696 of *Lecture Notes in Computer Science*, pages 419–430. Springer International Publishing, 2014.

[95] Yaping Luo, Jaap Stelma, and M. G. J van den Brand. Functional safety measurement in the automotive domain: Adaptation of psm. In *Proceedings of the First International Workshop on Automotive Software Architecture*, WASA '15, pages 11–17. ACM, 2015.

[96] Yaping Luo and M. G. J van den Brand. Metrics Design for Safety Assessment. *Information and Software Technology*, 2015. accepted. URL: `http://www.sciencedirect.com/science/article/pii/S0950584915002219`.

[97] Yaping Luo, M. G. J van den Brand, Luc Engelen, John Favaro, Martijn Klabbers, and Giovanni Sartori. Extracting Models from ISO 26262 for Reusable Safety Assurance. In *Safe and Secure Software Reuse*, pages 192–207. 2013.

[98] Yaping Luo, M. G. J van den Brand, Luc Engelen, and Martijn Klabbers. A Modeling Approach to Support Safety Certification in the Automotive Domain. In *FISITA 2014 World Automotive Congress*, 2014.

[99] Yaping Luo, M. G. J van den Brand, Luc Engelen, and Martijn Klabbers. A Modeling Approach to Support Safety Assurance in the Automotive Domain. In *Progress in Systems Engineering*, volume 1089, pages 339–345. Springer International Publishing, 2015.

[100] Yaping Luo, M. G. J van den Brand, and Alexandre Kiburse. Safety Case Development with SBVR-based Controlled Language. In *Proceedings of Third International Conference on Model-Driven Engineering and Software Development*, 2015.

[101] Yaping Luo, M. G. J van den Brand, and Zhuoao Li. A Categorization of GSN-based Safety Cases and Patterns. In *Proceedings of Fourth International Conference on Model-Driven Engineering and Software Development*, 2015.

[102] Yaping Luo, M.G.J van den Brand, Luc Engelen, and Martijn Klabbers. From Conceptual Models to Safety Assurance. In *Conceptual Modeling*, pages 195–208. Springer International Publishing, 2014.

[103] Helmut Martin, Martin Krammer, Bernhard Winkler, and Christian Schwarzl. Model-based Engineering Workflow for Automotive Safety Concepts. 2015.

[104] Yutaka Matsuno. D-Case Editor: A Typed Assurance Case Editor. *University of Tokyo*, 2011.

[105] John McGary, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Hall Fred. *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[106] Defence Standard 00-55 Part 1: Requirements for Safety Related Software in Defence Equipment, 1997. `http://www.software-supportability.org/Docs/00-55_Part_1.pdf`.

[107] Martin Monperrus, Jean-Marc Jézéquel, Benoit Baudry, Joël Champeau, and Brigitte Hoeltzener. Model-Driven Generative Development of Measurement Software. volume 10, pages 537–552. Springer, 2011.

[108] John Murdoch. Safety Measurement. *PSM Safety Measurement White Paper*, 2006.

[109] John Murdoch, Graham Clark, Antony Powell, and Paul Caseley. Measuring Safety: Applying PSM to the System Safety Domain. In *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, SCS '03, pages 47–55, 2003. URL: `http://dl.acm.org/citation.cfm?id=1082051.1082055`.

[110] Sunil Nair, Jose L. de la Vara, Alberto Melzi, Giorgio Tagliaferri, Laurent de la Beaujardiere, and Fabien Belmonte. Safety Evidence Traceability: Problem Analysis and Model. In *Requirements Engineering: Foundation for Software Quality*, volume 8396, pages 309–324, 2014.

[111] Anantha Narayanan, Tihamer Levendovszky, Daniel Balasubramanian, and Gabor Karsai. Automatic Domain Model Migration to Manage Metamodel Evolution. In *Model Driven Engineering Languages and Systems*, volume 5795, pages 706–711. 2009.

[112] Lina Nemuraite, Tomas Skersys, Algirdas Sukys, Edvinas Sinkevicius, and Linas Ablonskis. VETIS tool for editing and transforming SBVR business vocabularies and business rules into UML&OCL models. In *16th International Conference on Information and Software Technologies, Kaunas: Kaunas University of Technology*, pages 377–384, 2010.

[113] Paul B.F. Njonko and Walid El Abed. From Natural Language Business Requirements to Executable Models via SBVR. In *Systems and Informatics (ICSAI), 2012 International Conference on*, pages 2453–2457, 2012.

[114] OMG. Software and Systems Process Engineering Metamodel Specification, April 2008. http://www.omg.org/spec/SPEM/2.0/.

[115] OMG. MOF 2: XMI - Mapping Specification (Version 2.4.1). 2013. URL: http://www.omg.org/spec/XMI/2.4.1/.

[116] OMG. SBVR: Semantics Of Business Vocabulary And Rules (version 1.2), Sepetember 2013.

[117] OMG. Meta Object Facility (MOF) Specification v2.5, 2015. URL: http://www.omg.org/spec/MOF/2.5.

[118] Rob Palin, David Ward, Ibrahim Habli, and Roger Rivett. ISO 26262 Safety Cases: Compliance and Assurance. In *Proceedings of the 6th IET International Conference on System Safety*, pages 1–6, 2011.

[119] Jose Ignacio Panach, Sergio España, AnaM. Moreno, and Ócar Pastor. Dealing with Usability in Model Transformation Technologies. In *Conceptual Modeling*, volume 5231 of *Lecture Notes in Computer Science*, pages 498–511. Springer Berlin Heidelberg, 2008.

[120] RajwinderKaur Panesar-Walawege. *Using Model-Driven Engineering to Support the Certification of Safety-Critical Systems*. PhD thesis, University of Oslo, 2012.

[121] RajwinderKaur Panesar-Walawege, Mehrdad Sabetzadeh, and Lionel Briand. A Model-Driven Engineering Approach to Support the Verification of Compliance to Safety Standards. In *Software Reliability Engineering (ISSRE), 2011 IEEE 22nd International Symposium on*, pages 30–39, Nov 2011.

[122] RajwinderKaur Panesar-Walawege, Mehrdad Sabetzadeh, and Lionel Briand. Using Model-Driven Engineering for Managing Safety Evidence: Challenges, Vision and Experience. In *2011 First International Workshop on Software Certification (WoSoCER)*, pages 7–12. IEEE, Nov 2011.

[123] RajwinderKaur Panesar-Walawege, Mehrdad Sabetzadeh, and Lionel Briand. Using UML Profiles for Sector-Specific Tailoring of Safety Evidence Information. In Manfred Jeusfeld, Lois Delcambre, and Tok-Wang Ling, editors, *30th ACM International Conference on Conceptual Modeling (ER)*, volume 6998 of *Lecture Notes in Computer Science*, pages 362–378. Springer, Heidelberg, 2011.

[124] Chris Partridge, Cesar Gonzalez-Perez, and Brian Henderson-Sellers. Are Conceptual Models Concept Models? In *Conceptual Modeling*, volume 8217 of *Lecture Notes in Computer Science*, pages 96–105. Springer Berlin Heidelberg, 2013.

[125] Zvezdan Protić. *Configuration Management for Models: Generic Methods for Model Comparison and Model Co-evolution*. PhD thesis, Eindhoven University of Technology, 2011.

[126] Safety Case Repository, 2013. http://dependability.cs.virginia.edu/info/Safety_Cases:Repository.

[127] Norbert FM Roozenburg and Johannes Eekels. *Product Design: Fundamentals and Methods*, volume 2. Wiley Chichester, 1995.

[128] Alejandra Ruiz, Ibrahim Habli, and Huáscar Espinoza. Towards a Case-Based Reasoning Approach for Safety Assurance Reuse. In Frank Ortmeier and Peter Daniel, editors, *Computer Safety, Reliability, and Security*, volume 7613 of *Lecture Notes in Computer Science*, pages 22–35. Springer, Heidelberg, 2012.

[129] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples.* John Wiley & Sons, 2012.

[130] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack. The Epsilon Object Language (EOL). In *Model Driven Architecture–Foundations and Applications*, pages 128–142. Springer, 2006.

[131] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A.C. Polack. The Epsilon Transformation Language. In *Theory and practice of model transformations*, pages 46–60. Springer, 2008.

[132] Dimitrios S. Kolovos, Louis M. Rose, Saad Bin Abid, Richard F. Paige, Fiona A.C. Polack, and Goetz Botterweck. Taming EMF and GMF Using Model Transformation. In *Proceedings of MODELS'10*, 2010.

[133] Arash Khabbaz Saberi, Yaping Luo, Filip Pawel Cichosz, M. G. J van den Brand, and Sven Jansen. An Approach for Functional Safety Improvement of an Existing Automotive System. In *8th Annual IEEE System Conference*, pages 277–282, 2015.

[134] SAE. Arp4754: Certification considerations for highly-integrated or complex aircraft systems. *SAE, Warrendale, PA*, 1996.

[135] Iván Santiago, Juan Manuel Vara, María Valeria de Castro, and Esperanza Marcos. Towards the Effective Use of Traceability in Model-Driven Engineering Projects. In *Conceptual Modeling*, pages 429–437. 2013.

[136] David J Smith and Kenneth GL Simpson. *Safety Critical Systems Handbook. A Straightforward Guide to Functional Safety, IEC 61508 (2010 Edition) and Related Standards, Including Process IEC 61511 and Machinery IEC 62061 and ISO 13849.* Elsevier, 2010.

[137] Silvie Spreeuwenberg and Keri Anderson Healy. SBVR's Approach to Controlled Natural Language. In *Proceedings of the 2009 conference on Controlled natural language*, CNL'09, pages 155–169, Berlin, Heidelberg, 2010. Springer-Verlag.

[138] Dean H Stamatis. *Failure mode and effect analysis: FMEA from theory to execution.* ASQ Quality Press, 2003.

[139] Patrik Sternudd. Unambiguous Requirements in Functional Safety and ISO 26262: Dream or Reality? Master's thesis, Uppsala University, 2011.

[140] David C. Sutton. Linguistic problems with requirements and knowledge elicitation. volume 5, pages 114–124. Springer-Verlag London Limited, 2000.

[141] Bernhard Thalheim. The Theory of Conceptual Models, the Theory of Conceptual Modelling and Foundations of Conceptual Modelling. In David W. Embley and Bernhard Thalheim, editors, *Handbook of Conceptual Modeling*, pages 543–577. Springer Berlin Heidelberg, 2011.

[142] Bernhard Thalheim. The Science and Art of Conceptual Modelling. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems VI*, pages 76–105. Springer, 2012.

[143] OMG UML. 2.0 Superstructure Specification. 2004.

[144] M. G. J van den Brand. Model-Driven Engineering Meets Generic Language Technology. In *Software Language Engineering*, volume 5452 of *LNCS*, pages 8–15. 2009.

[145] JoseLuis Vara and RajwinderKaur Panesar-Walawege. SafetyMet: A Metamodel for Safety Standards. In *Model-Driven Engineering Languages and Systems*, volume 8107 of *Lecture Notes in Computer Science*, pages 69–86. Springer Berlin Heidelberg, 2013.

[146] Guido Wachsmuth. Metamodel Adaptation and Model Co-adaptation. In *ECOOP 2007 - Object-Oriented Programming*, volume 4609, pages 600–624. 2007.

[147] David D. Ward and Steve E. Crozier. The Uses and Abuses of ASIL Decomposition in ISO 26262. In *7th IET International Conference on System Safety, incorporating the Cyber Security Conference*, pages 1–6, Oct 2012.

[148] Katarzyna Warszawska and Andrzej Kraslawski. Method for Quantitative Assessment of Safety Culture. Elsevier, 2015.

[149] Rob Weaver, Georgios Despotou, Tim Kelly, and John McDermid. Combining Software Evidence: Arguments and Assurance. volume 30, pages 1–7. May 2005.

[150] John R Wilson and Sarah Sharples. *Evaluation of Human Work*. CRC Press, 2015.

[151] S.P. Wilson, Tim Kelly, and John McDermid. Safety Case Development: Current Practice, Future Prospects. In Roger Shaw, editor, *Safety and Reliability of Software Based Systems*, pages 135–156. Springer London, 1997.

[152] Ji Wu, Tao Yue, S. Ali, and Huihui Zhang. Ensuring Safety of Avionics Software at the Architecture Design Level: An Industrial Case Study. In *13th International Conference on Quality Software (QSIC)*, pages 55–64, July 2013.

[153] Gregory Zoughbi, Lionel Briand, and Yvan Labiche. Modeling Safety and Airworthiness (RTCA DO-178B) Information: Conceptual Model and UML Profile. volume 10, pages 337–367. Springer-Verlag New York, Inc., Secaucus, NJ, USA, July 2011.

Appendix A

Table of Content of ISO 26262 Part 3

Summary

From Conceptual Model to Safety Assurance

Applying Model-Based Techniques to Support Safety Assurance

In safety-critical domains such as automotive, railway, and avionics, even a small failure of a system might cause injury or death to people. A number of international safety standards are introduced as guidelines for system suppliers to keep the risk of systems at an acceptable level. Those standards are typically large documents containing a huge number of requirements for system development. The safety standards describe generalized approaches to identifying hazards and risks, design life-cycles, and analysis and design techniques. Therefore, when applying such standards for a specific application, significant degree of interpretation of those standards may be necessary.

The process for developing safety-critical systems in these safety domains is manually checked for compliance with the standards. This checking process is referred as safety assurance and certification. Due to the amount of manual work involved, safety assurance is usually costly and time-consuming. Moreover, when a system evolves, some of the existing safety-assurance data needs to be regathered or re-validated. To address this, we started our research on safety *standard-based approaches*. We have proposed a rule-based approach to model the ISO 26262 standard. For demonstration, the model of ISO 26262 Part 3 has been extracted and validated by domain experts. Moreover by utilizing metamodel transformation, we have provided an approach to drive domain or project specific metamodels using a generic metamodel as basis. The companies in safety-critical domains could not only use the generic metamodel for sharing patterns of certification assessment, but also keep their way of working by using their domain concepts. Finally, to facilitate these companies to find the reusable data from models conforming to similar metamodels, the metamodel comparison and traceability management during the metamodel transformation have also been discussed.

In some safety standards, safety case development is highly recommend to justify the safety of a system. The Goal Structuring Notation (GSN) provides a graphical way to construct a safety case. However, the content of the safety case elements, such as safety claims, is in natural language. Therefore, a common understanding of the meaning of a safety claim may be difficult to reach. Consequently, the confidence of a safety claim can be misplaced. Based on these observations, we have carried out our study on safety *argument-based approaches*. We have proposed to use an Semantics of Business Vocabulary and Business Rules (SBVR) based controlled language to support safety case development. By using the controlled language, the ambiguities caused by natural language can be mitigated. We have also developed an SBVR editor for building a vocabulary and a GSN

editor with vocabulary support. Furthermore, the SBVR safety claims can also be used to facilitate safety evidence collection.

Finally the research on the *overall safety assessment process* in the automotive domain has been carried out. The ISO 26262 standard has been studied to facilitate functional safety management for an existing system. Moreover, by studying on the whole process of the safety assessment, we have observed that this process is hard to be estimated due to the manual work. To monitor safety assessment process, for instance, identify costly activities, a methodology has been proposed to design metrics for safety assessment from three different perspectives: industrial interests, safety standards, available data. The metrics can help stakeholders, for example safety managers, to estimate the overall cost and monitor the whole compliance process. Besides, the results of these metrics can also help them make decisions during safety assurance process.

To summarize, our research in this thesis consists of three main parts: standard-based approaches, argument-based approaches, approaches on overall safety assessment process. The standard-based approaches discussed how to use metamodeling techniques to obtain standard conceptual models/metamodels. The argument-based approaches introduced how to use controlled-language to facilitate safety argument construction and safety evidence collection. The approaches on overall safety assessment process described how to apply the ISO 26262 standard for an existing system and how to extract metrics for the safety assessment process. Although we mainly focused on the automotive domain in this thesis, our approaches can be applied to other safety-critical domains as well.

Curriculum Vitae

Personal Information

Name: Yaping Luo

Date of birth: November 15, 1987

Place of birth: Henan, China

Education

Ph.D Candidate *2012–now*

Eindhoven University of Technology

Eindhoven, the Netherlands

M.Sc in Computer Science and Engineering *2009–2012*

Beihang University

Beijing, China

B.Sc in Computer Science and Engineering *2005–2009*

Beihang University

Beijing, China

Research Project Experience

FP7 OPENCOSS project (an Open Platform for EvolutioNary Certification Of Safety-critical Systems) *2012.2–2015.3*

Eindhoven University of Technology

Eindhoven, the Netherlands

ITEA2 DiYSE: Do-it-Yourself Smart Experiences European Project (code: 08005) 2011.3–2011.8

Universidad Politécnica de Madrid (UPM),

Madrid, Spain

BUAA Innovative Funding Project on Wireless Sensor Networks QoS and Control Technology *2010.12–2011.3*

Beihang University

Beijing, China

China's Natural Science Foundation (60803120) and Doctoral Fund of Ministry of Education of China (20091102110017) research group *2009.10–2010.12*

Beihang University

Beijing, China

IPA Dissertation Series

Titles in the IPA Dissertation Series since 2013

H. Beohar. *Refinement of Communication and States in Models of Embedded Systems.* Faculty of Mathematics and Computer Science, TU/e. 2013-01

G. Igna. *Performance Analysis of Real-Time Task Systems using Timed Automata.* Faculty of Science, Mathematics and Computer Science, RU. 2013-02

E. Zambon. *Abstract Graph Transformation – Theory and Practice.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-03

B. Lijnse. *TOP to the Rescue – Task-Oriented Programming for Incident Response Applications.* Faculty of Science, Mathematics and Computer Science, RU. 2013-04

G.T. de Koning Gans. *Outsmarting Smart Cards.* Faculty of Science, Mathematics and Computer Science, RU. 2013-05

M.S. Greiler. *Test Suite Comprehension for Modular and Dynamic Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-06

L.E. Mamane. *Interactive mathematical documents: creation and presentation.* Faculty of Science, Mathematics and Computer Science, RU. 2013-07

M.M.H.P. van den Heuvel. *Composition and synchronization of real-time components upon one processor.* Faculty of Mathematics and Computer Science, TU/e. 2013-08

J. Businge. *Co-evolution of the Eclipse Framework and its Third-party Plug-ins.* Faculty of Mathematics and Computer Science, TU/e. 2013-09

S. van der Burg. *A Reference Architecture for Distributed Software Deployment.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2013-10

J.J.A. Keiren. *Advanced Reduction Techniques for Model Checking.* Faculty of Mathematics and Computer Science, TU/e. 2013-11

D.H.P. Gerrits. *Pushing and Pulling: Computing push plans for disk-shaped robots, and dynamic labelings for moving points.* Faculty of Mathematics and Computer Science, TU/e. 2013-12

M. Timmer. *Efficient Modelling, Generation and Analysis of Markov Automata.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2013-13

M.J.M. Roeloffzen. *Kinetic Data Structures in the Black-Box Model.* Faculty of Mathematics and Computer Science, TU/e. 2013-14

L. Lensink. *Applying Formal Methods in Software Development.* Faculty of Science,

Mathematics and Computer Science, RU. 2013-15

C. Tankink. *Documentation and Formal Mathematics — Web Technology meets Proof Assistants.* Faculty of Science, Mathematics and Computer Science, RU. 2013-16

C. de Gouw. *Combining Monitoring with Run-time Assertion Checking.* Faculty of Mathematics and Natural Sciences, UL. 2013-17

J. van den Bos. *Gathering Evidence: Model-Driven Software Engineering in Automated Digital Forensics.* Faculty of Science, UvA. 2014-01

D. Hadziosmanovic. *The Process Matters: Cyber Security in Industrial Control Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-02

A.J.P. Jeckmans. *Cryptographically-Enhanced Privacy for Recommender Systems.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-03

C.-P. Bezemer. *Performance Optimization of Multi-Tenant Software Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2014-04

T.M. Ngo. *Qualitative and Quantitative Information Flow Analysis for Multi-threaded Programs.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-05

A.W. Laarman. *Scalable Multi-Core Model Checking.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-06

J. Winter. *Coalgebraic Characterizations of Automata-Theoretic Classes.* Faculty of Science, Mathematics and Computer Science, RU. 2014-07

W. Meulemans. *Similarity Measures and Algorithms for Cartographic Schematization.* Faculty of Mathematics and Computer Science, TU/e. 2014-08

A.F.E. Belinfante. *JTorX: Exploring Model-Based Testing.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2014-09

A.P. van der Meer. *Domain Specific Languages and their Type Systems.* Faculty of Mathematics and Computer Science, TU/e. 2014-10

B.N. Vasilescu. *Social Aspects of Collaboration in Online Software Communities.* Faculty of Mathematics and Computer Science, TU/e. 2014-11

F.D. Aarts. *Tomte: Bridging the Gap between Active Learning and Real-World Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2014-12

N. Noroozi. *Improving Input-Output Conformance Testing Theories.* Faculty of Mathematics and Computer Science, TU/e. 2014-13

M. Helvensteijn. *Abstract Delta Modeling: Software Product Lines and Beyond.* Faculty of Mathematics and Natural Sciences, UL. 2014-14

P. Vullers. *Efficient Implementations of Attribute-based Credentials on Smart Cards.* Faculty of Science, Mathematics and Computer Science, RU. 2014-15

F.W. Takes. *Algorithms for Analyzing and Mining Real-World Graphs.* Faculty of Mathematics and Natural Sciences, UL. 2014-16

M.P. Schraagen. *Aspects of Record Linkage.* Faculty of Mathematics and Natural Sciences, UL. 2014-17

G. Alpár. *Attribute-Based Identity Management: Bridging the Cryptographic Design of ABCs with the Real World.* Faculty of Science, Mathematics and Computer Science, RU. 2015-01

A.J. van der Ploeg. *Efficient Abstractions for Visualization and Interaction.* Faculty of Science, UvA. 2015-02

R.J.M. Theunissen. *Supervisory Control in Health Care Systems.* Faculty of Mechanical Engineering, TU/e. 2015-03

T.V. Bui. *A Software Architecture for Body Area Sensor Networks: Flexibility and Trustworthiness.* Faculty of Mathematics and Computer Science, TU/e. 2015-04

A. Guzzi. *Supporting Developers' Teamwork from within the IDE.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-05

T. Espinha. *Web Service Growing Pains: Understanding Services and Their Clients.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-06

S. Dietzel. *Resilient In-network Aggregation for Vehicular Networks.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-07

E. Costante. *Privacy throughout the Data Cycle.* Faculty of Mathematics and Computer Science, TU/e. 2015-08

S. Cranen. *Getting the point — Obtaining and understanding fixpoints in model checking.* Faculty of Mathematics and Computer Science, TU/e. 2015-09

R. Verdult. *The (in)security of proprietary cryptography.* Faculty of Science, Mathematics and Computer Science, RU. 2015-10

J.E.J. de Ruiter. *Lessons learned in the analysis of the EMV and TLS security protocols.* Faculty of Science, Mathematics and Computer Science, RU. 2015-11

Y. Dajsuren. *On the Design of an Architecture Framework and Quality Evaluation for Automotive Software Systems.* Faculty of Mathematics and Computer Science, TU/e. 2015-12

J. Bransen. *On the Incremental Evaluation of Higher-Order Attribute Grammars.* Faculty of Science, UU. 2015-13

S. Picek. *Applications of Evolutionary Computation to Cryptology.* Faculty of Science, Mathematics and Computer Science, RU. 2015-14

C. Chen. *Automated Fault Localization for Service-Oriented Software Systems.* Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2015-15

S. te Brinke. *Developing Energy-Aware Software.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-16

R.W.J. Kersten. *Software Analysis Methods for Resource-Sensitive Systems.* Faculty of Science, Mathematics and Computer Science, RU. 2015-17

J.C. Rot. *Enhanced coinduction.* Faculty of Mathematics and Natural Sciences, UL. 2015-18

M. Stolikj. *Building Blocks for the Internet of Things.* Faculty of Mathematics and Computer Science, TU/e. 2015-19

D. Gebler. *Robust SOS Specifications of Probabilistic Processes.* Faculty of Sciences, Department of Computer Science, VUA. 2015-20

M. Zaharieva-Stojanovski. *Closer to Reliable Software: Verifying functional behaviour of concurrent programs.* Faculty of Electrical Engineering, Mathematics & Computer Science, UT. 2015-21

R.J. Krebbers. *The C standard formalized in Coq.* Faculty of Science, Mathematics and Computer Science, RU. 2015-22

R. van Vliet. *DNA Expressions – A Formal Notation for DNA.* Faculty of Mathematics and Natural Sciences, UL. 2015-23

S.-S.T.Q. Jongmans. *Automata-Theoretic Protocol Programming.* Faculty of Mathematics and Natural Sciences, UL. 2016-01

S.J.C. Joosten. *Verification of Interconnects*. Faculty of Mathematics and Computer Science, TU/e. 2016-02

M.W. Gazda. *Fixpoint Logic, Games, and Relations of Consequence*. Faculty of Mathematics and Computer Science, TU/e. 2016-03

S. Keshishzadeh. *Formal Analysis and Verification of Embedded Systems for Healthcare*. Faculty of Mathematics and Computer Science, TU/e. 2016-04

P.M. Heck. *Quality of Just-in-Time Requirements: Just-Enough and Just-in-Time*. Faculty of Electrical Engineering, Mathematics, and Computer Science, TUD. 2016-05

Y. Luo. *From Conceptual Models to Safety Assurance – Applying Model-Based Techniques to Support Safety Assurance*. Faculty of Mathematics and Computer Science, TU/e. 2016-06