Systematic engineering of safe open adaptive systems shown for truck platooning

Vom Fachbereich Informatik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Master of Science (M. Sc.)

genehmigte Masterarbeit
von

Jan Reich

Datum:	15.12.2016
Erstprüfer:	Prof. Dr.-Ing. habil. Peter Liggesmeyer
Zweitprüfer:	Dr. Daniel Schneider
Betreuer:	Dr. Rasmus Adler

Fachbereich Informatik, Technische Universität Kaiserslautern
Fraunhofer-Institut für Experimentelles Software Engineering (IESE)

Selbstständigkeitserklärung

Hiermit erkläre ich, Jan Reich, dass ich die vorliegende Masterarbeit mit dem Thema

„Systematic engineering of safe open adaptive systems shown for truck platooning“

selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Kaiserslautern, den 15.12.2016

Jan Reich

Abstract

Recently, conditional safety certificates (ConSerts) have been proposed as a novel means for the safety assurance of collaborations between open adaptive systems. Since the innovation potential of such collaborations has been recognized in many different application domains, it deemed desirable to demonstrate the general applicability of the ConSert approach across multiple domains. As yet, feasibility studies have only been conducted in the agricultural and ambient-assisted living domains showing that ConSerts can in principle assure a sufficient level of safety for collaborations in these domains. Therefore, this thesis addresses the application and evaluation of the ConSert approach in another case study in the automotive domain considering a platooning collaboration between two trucks, where an autonomously operated truck should automatically follow a human-driven truck leading the platoon. To enable ConSerts to constrain the effects of safety-critical system behavior deviations on the collaboration, a description of the collaboration's intended behavior is required as a necessary prerequisite for the definition of concrete behavior deviations. Thus, a detailed approach for the systematic construction of related engineering models has been developed that resulted in a service-oriented description of the intended platooning behavior being suitable for the subsequent methodological derivation of ConSert models. In order to provide guarantees assuring that platooning is safe even during the presence of certain failure-caused behavior deviations, a simulative approach has been utilized based on a realistic platooning simulation model. The evaluation of the case study executed in this thesis has yielded two primary results: On the one hand, evidence has been provided that the ConSert approach can be successfully applied for platooning collaborations in the automotive domain. On the other hand, the proposed method for the specification of safe intended behaviors for collaborations has been defined to cover a wider scope leaving enough way to easily adapt it for similar application scenarios.

Kurzfassung

Erst kürzlich wurden „Conditional Safety Certificates“ (ConSerts) als eine neuartige Methode vorgeschlagen, um die funktionale Sicherheit von Kollaborationen zwischen offenen adaptiven Systemen zu gewährleisten. Da das Innovationspotenzial solcher Kollaborationen in vielen unterschiedlichen Anwendungsdomänen bereits erkannt wurde, erschien es wünschenswert, die generelle Anwendbarkeit des ConSert-Ansatzes in solch vielfältigen Domänen zu demonstrieren. Bisher wurden lediglich Machbarkeitsstudien in der Agrardomäne sowie in der Domäne des umgebungsunterstützen Lebens durchgeführt mit dem Ergebnis, dass ConSerts prinzipiell in der Lage sind, funktionale Sicherheit für Kollaborationen in diesen Anwendungsbereichen zu gewährleisten. Daher beschäftigt sich die vorliegende Arbeit mit der Anwendung und Evaluierung des ConSert-Ansatzes in einer weiteren Fallstudie aus dem Automobilbereich, welche die Kollaboration in einer Kolonnenfahrt zweier Lkw zum Gegenstand hat, bei der ein autonom gesteuerter Lkw automatisch einem von einem Fahrer gesteuerten Lkw folgt. Um ConSerts zu ermöglichen, die Effekte von sicherheitskritischen Verhaltensabweichungen der Systeme auf die Kollaboration einzuschränken, ist eine Beschreibung des beabsichtigen Verhaltens der Kollaboration Vorbedingung für die Definition konkreter Verhaltensabweichungen. Zu diesem Zweck wurde eine detaillierter Ansatz zur systematischen Erstellung der zugehörigen Entwicklungsmodelle erarbeitet, welcher es ermöglicht, durch eine serviceorientierte Spezifikation der beabsichtigten Kollaboration eine geeignete Basis für die methodische Herleitung von ConSert Modellen zu schaffen. Ein Simulationsansatz auf Basis eines realistischen Simulationsmodells für die Kolonnenfahrt ermöglichte eine ausreichende Validierung, um letztlich Garantien für die Gewährleistung einer sicheren Kolonnenfahrt unter dem Einfluss von durch Fehler verursachten Verhaltensabweichungen aussprechen zu können. Die Evaluation der Fallstudie lieferte zwei primäre Resultate: Einerseits wurde gezeigt, dass der Consert-Ansatz für Kolonnen-Kollaborationen im Automobilbereich erfolgreich angewandt werden kann. Andererseits wurde die vorgeschlagene Methode zur Erstellung der Spezifikation eines sicheren Normalverhaltens von Kollaborationen so weit gefasst, dass sie ohne Schwierigkeiten für ähnliche Anwendungen angepasst werden kann.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the last two decades, society witnessed a strong trend towards a high level of connectivity, which can be observed by looking at how smartphones found their way into our daily lives. Nowadays, they provide useful services that can significantly improve how human-beings manage their daily routines. Examples are automatic traffic alerts based on the detection of our typical movement habits or guidance in foreign places where we are not street-smart.

The essential ingredients of such functionalities are on the one hand the connectivity of devices and on the other hand the composition of multiple separate services into higher-level emergent services providing an innovative value to their consumers.

Apart from the private sector, various commercial application domains have discovered the potential of connectivity and service composition, too. The automotive domain leads the way in researching use cases where cars communicate with other cars and smart infrastructures (Car2X) while driving and thus providing emergent services to the drivers. These include for example systems that can significantly increase safety by warning other road users of obstacles or traffic jams long before they arrive at the respective places. Thinking further, if complete vehicle fleets were interconnected, their data could be used to make traffic and road management much more efficient by individual and dynamic navigation of cars. This type of application is referred to as *smart mobility* in literature and media [1]. Additionally considering the simultaneously increasing level of autonomy, human drivers can be relieved from their driving task in the long run and can focus on other activities while still maintaining high levels of safety and driving comfort.

It can be expected that the automotive domain and in particular the commercial vehicle sector will be highly impacted in a positive sense by the trend towards more autonomy and Car2X communication. A McKinsey study on the future of the commercial vehicle sector shows that the global market turnover will increase by almost 50% until 2025 [2]. The identified main drivers for this increase are autonomy as well as inter-vehicle connectivity, which demonstrates the importance of these concepts for research, too.

Many of these conceivable use cases that could be realized in the future, are typically dependent on both collaborative knowledge and actions performed by interconnected networks of systems, so called systems of sys-

tems. In literature, systems of systems are typically also referred to as cooperative systems, cyber-physical systems or – like in this thesis – open adaptive systems (OAS). These kinds of systems are open in that they are designed to collaborate at runtime with other potentially unknown systems. In addition, they should be able to adapt themselves dynamically to their given environmental context in a preferably optimal way.

1.2 Problem statement

Since a lot of innovative collaborations of systems are inherently safety-critical, i.e. they might cause harm or damage to their environment, the assurance of a sufficient level of safety is a high priority objective for the development of these systems.

Guaranteeing the required safety level however is a serious challenge, because traditional safety engineering and assurance approaches are usually not applicable without further ado. Traditional safety engineering approaches only target closed systems. Thus, they can leverage from a complete knowledge about the system's structure, interaction schemes and environment at design time. Having this knowledge, safety experts typically make use of model-driven approaches that enable a modular safety assurance of the system components and their assemblies. State-of-the-art realizations of model-driven safety approaches cover all activities of the safety life cycle, namely hazard analysis and risk assessment [3], safety analysis [4] and safety requirement specification. Finally, a safety case assembles the results of the former activities into one central model serving as an argumentation basis for the final certification step.

In the context of OAS, traditional approaches are hardly applicable because essential information about the structure and interactions between the systems at runtime is missing at design time. This includes the lack of knowledge on how the services provided by a system or one of its components will be consumed by other potentially unknown systems at runtime. In addition, both openness and adaptivity aspects foster the reuse of systems or their components in new OAS. The consequence is that system developers can hardly foresee how the functionality of their system under development might be (mis)used in the future and thus how the implemented variation points of the system will be resolved at runtime.

A general strategy to overcome this problem is to shift safety engineering activities from design time to runtime. At runtime, all necessary information for taking informed decisions with respect to safety is available. The shifting of the activities however requires the system itself to act more intelligent by being aware of its own safety properties during operation. This exceeds the capabilities of simple fault tolerance mechanisms, because these only preserve correct service in the presence of active system-internal faults, but do not have to consider all conceivable kinds of collaboration partners and their behaviors.

If safety engineering activities are shifted to runtime, their models must also be available at runtime. This imposes additional requirements with respect to the models' machine-readable representations in terms of space and execution time efficiency, because especially commercial embedded hardware is typically dimensioned rather concise concerning the available memory space and computing power. Apart from hardware requirements, machine-readable models require a certain degree of formalism to be processed solely by a computer.

A promising approach that follows the strategy of shifting certain safety activities to runtime is given by conditional safety certificates (ConSerts) [5]. Evaluation studies of this approach have already been conducted in the agricultural as well as the ambient assisted living domains. The studies have shown that the ConSert approach is capable of providing the conceptual framework for enabling safety assurance of OAS in general and of tractor-implement automation scenarios as well as health emergency detection scenarios in particular.

However, assuring safety for open adaptive systems based on ConSerts is still a challenging task for two reasons:

1. It is not known yet, if the ConSert approach is in general sufficient to cover any application domain and application setting.
2. There is neither an established engineering approach nor a detailed guideline for the application of the ConSert approach.

In the following section, the contribution of this thesis regarding the solution to the above mentioned problems will be stated.

1.3 Thesis goals

This thesis aims at contributing to the solution of the challenges being identified in the problem statement with respect to the application and evaluation of the ConSert approach. In order to tackle the first identified problem, the approach should be applied and evaluated within another case study for the automotive domain.

The automotive domain differs from the agricultural domain, for which ConSerts have been evaluated already, in that it is driven by different regulatory standards, which directly affect how OAS have to be developed for a specific domain. Furthermore, the runtime environment of automotive applications is typically less constrained, e.g. there are potentially more different kinds of collaboration partners the OAS has to deal with.

As a concrete application scenario, truck platooning has been chosen. In a nutshell, truck platooning is an operational mode, where multiple trucks on a highway drive together in a convoy, also called platoon, by maintaining very small inter-vehicular distances. The main benefit of platooning is a

reduced amount of needed fuel for all participating trucks resulting from a reduction of aerodynamic drag when close spacing is maintained. In addition, close spacing yield a better space occupation efficiency on highways in general.

Note that truck platooning also represents a different application setting than the already evaluated tractor-implement scenario in the agricultural domain. When tractors and implements should be integrated to safely collaborate on the field at runtime, their compatibility with respect to safety is checked only once at that time. Afterwards, they will be *mechanically* coupled and thus their connection will be static during collaboration. In contrast, the truck platooning scenario includes a higher degree of dynamics in that first, the trucks are only coupled through a wireless connection and second, the environmental influences like road or weather conditions as well as the potential diversities of the trucks themselves have a much more dynamic impact on the collaboration.

The differences with respect to application domain and application setting make truck platooning an interesting scenario for an evaluation of the ConSert approach. In order to make such a platooning collaboration safe for all participating trucks, two different aspects of its development will be considered concretely in this thesis:

Goal 1 – Systematic engineering of safe nominal behavior for the collaboration between two trucks

The main functional goal of truck platooning is to maintain a distance between the trucks that is as short as possible but still safe. "Safe" in this context means that certain critical accidents like a frontal crash cannot occur, for instance when the leader truck has higher braking capabilities than the follower truck and performs an emergency brake maneuver so that the separation distance is not able to compensate for this diversity. The accident in this example is neither caused by a systematic software nor a random hardware failure but caused by the intended functionality itself, namely the fact that the maintained separation was too small with respect to the given collaboration context factors. This shows that the safety of the intended collaboration, which will be referred to as *safe nominal behavior* throughout this thesis, needs to be considered already in the functional specification of the OAS.

In terms of truck platooning the only way to reliably determine and maintain a safe distance between the trucks is through the mutual exchange of information about the trucks' motion states as well as the collaboration's environmental context such as road conditions at runtime. Thus, it is necessary to decide and formalize in the functional specification of the collaboration,

- which information needs to be exchanged to achieve the collaboration goal,

- how often that information needs to be exchanged between the collaboration partners and
- which collaboration partner has access to the required information

In general, an information typically runs through the following temporal sequence within an OAS: analog-to-digital conversion in sensors, sensor post-processing such as noise filtering, fixed-point or floating-point computations, transformations into and from different memory representations, de-/modulation on a wireless medium and finally its feedback into the environment in shape of a set point for an actuator. All of these steps have in common that they take a certain processing time and more importantly that they affect the quality of the information either in a positive (e.g. sensor fusion) or in a negative way (e.g. rounding errors during floating-point computations). Thus, it is in particular important to know for the receiver of an information to which extent the information reflects the reality. This can be achieved by enriching an information with an indication about its quality, which could be for example value confidence intervals or timing guarantees.

In summary, this goal contains the creation of a safe nominal behavior specification for the truck platooning scenario, which forms the basis for analyzing failure-caused deviations from that behavior by specifying ConSert models.

Goal 2 – Methodological derivation and construction of related ConSerts models for guaranteeing a safe collaboration during failure

Having specified a proper safe nominal behavior specification for truck platooning, accidents having their cause in the intended functionality cannot occur anymore. However, it can be expected that either random hardware or systematic software faults are existent in platooning-relevant parts of both trucks and eventually lead to failures which themselves propagate through the platooning system. We need to distinguish between two types of these failure propagations:

Firstly, failures can propagate through the system of one collaboration partner, over which complete knowledge is existent at design time from the point of view of a system manufacturer. In the platooning scenario, the scope of such failure propagations would be one truck only and thus they can be mitigated by means of classical functional safety within that truck.

Secondly, failures in shape of faulty values may also propagate through the collaboration interface to another truck. In order to still maintain a safe collaboration, the receiving truck can tolerate faulty values to a certain degree while "paying" for this tolerance with an increased distance that is non-optimal with respect to fuel-saving. Another strategy is to decide to dissolve the platoon for safety reasons. In any case, the bounds of the

tolerated deviations need to be quantified and communicated so that the receiving truck can react properly.

ConSerts provide a methodological framework for formally specifying the bounds of deviations from the nominal behavior in shape of guarantees and demands. Each truck as a collaboration partner has to specify on the one hand, which type of information it can provide with what accuracy. However, the range of this accuracy guarantee can change, e.g. due to failures. On the other hand, the truck also poses demands to its collaboration partners indicating, how much deviation from the nominal behavior it can tolerate while still guaranteeing a safe collaboration. In order to finally achieve a *safe fail behavior* for platooning at runtime, the demands of all trucks have to be matched by guarantees of other trucks.

In summary, this goal deals with the operationalization of ConSerts for the truck platooning scenario. The construction of ConSerts includes activities and artifacts that need to be aligned with typical and suitable safety engineering methods and techniques. Furthermore, they need to be conceptualized with respect to their integration into safety engineering tools to leverage their potential for (semi-)automation. In addition, it should be analyzed how ConSerts can support a dynamic optimization of business-related properties with respect to changing environmental conditions or truck variabilities.

1.4 Thesis structure

This thesis is structured in seven chapters. Following this introduction, *Chapter 2* provides an overview of the foundations of open adaptive systems and the state of the art of their safety assurance. In addition, safety-related research for truck platooning is given. After having introduced the running example in more detail at the beginning of *Chapter 3*, a solution overview is presented exemplifying the thesis goals as set out in *Section 1.3* using the running example and presents both the respective challenges and how they can be addressed by existing methods. Next, *Chapter 4* presents the solution approach for the first goal by elaborating on the specific activities and artifacts that, being followed, yield a safe nominal behavior specification. *Chapter 5* builds upon its preceding chapter in that it explains a solution for the second goal, namely how to construct a safe fail behavior for truck platooning while tolerating certain kinds of deviations from the nominal behavior. Afterwards, *Chapter 6* first gives a summary followed by an evaluation to which extent the thesis goals have been met and a critical discussion on the results as well as lessons learned during the case study execution. Finally, a conclusion and possible areas of future work are presented.

2 Related work

This chapter presents the foundations on which the work of this thesis is built on. Since this thesis aims at the application and evaluation of the ConSert approach for open adaptive systems (OAS), *Sections 2.1* and *2.2* first describe the state of the art of OAS as well as the ConSert approach. Next, *Section 2.3* addresses how vehicle platooning has been realized in the past and what efforts have been spent to assure its safety.

2.1 Open adaptive systems

In order to understand which complexities arise during the transition from the development of closed systems towards the development of OAS this section will introduce a basic definition of OAS as well a distinction between closed systems and OAS. In addition, it will be described how the concepts of *service-oriented architectures* (SOA) have been applied to OAS and more specifically to automotive OAS in the past.

2.1.1 Definition of OAS

Disregarding the adaptivity aspect at first, some definitions for open systems of systems (SoS), sometimes also referred to as cyber-physical systems or cooperative systems, will be given in this subsection. The results of an extensive literature survey on SoS have been published in [6] and [7]. These results contain numerous definitions from literature, from which only those are presented here, which the author deemed best applicable for the scope of this thesis, namely systems of *embedded* systems.

Definition 1 SoS exist when there is a presence of a majority of the following five characteristics: operational and managerial independence, geographic distribution, emergent behavior and evolutionary development.

Definition 2 SoS engineering involves the integration of systems into SoS that ultimately contribute to the evolution of the social infrastructure.

Definition 3 SoS are large-scale concurrent and distributed systems that are comprised of complex systems.

Definition 4 SoS are large-scale integrated systems which are heterogeneous and independently operable on their own, but are networked for a common goal. The goal may be cost, performance, robustness, etc.

The favorite definition of this thesis' author for SoS is given in [8], because it clearly distinguishes the role of single systems and their composition into SoS as well as the creation of a value that cannot be achieved by single systems alone.

Definition 5 A SoS is comprised of autonomous constituent systems fulfilling an objective as independent systems but beyond that interdependent via interoperability to fulfill holistic objectives. A commonly discussed set of characteristics are autonomy, belonging, connectivity, diversity and emergence as they contrast a system and a SoS. SoS emergent behavior and capabilities is a product of the interactions that are greater than the sum of the independent actions of the constituent elements describing a constitutive SoS.

The five characteristics autonomy, belonging, connectivity, diversity and emergence describe the core sources for challenges that have to be addressed in the systems engineering discipline during the ongoing transition from single closed systems to SoS.

The term **open adaptive system (OAS)** that will be used throughout this thesis to describe the *collaborating single systems* within SoS, emphasizes the connectivity and diversity aspects of *Definition 5*, because these are the main challenges being addressed by the ConSert approach in assuring safety for SoS.

2.1.2 Transition from closed to open adaptive systems

The essential nature of closed embedded systems is that they are controlling physical processes based on perceiving their environment through sensors and transforming these inputs through their functionality into outputs affecting the environment in the desired way. Thus, the system's boundary structure and its interactions with the environment are known at design time. Since the only existing sources of knowledge are sensors and the system itself, the space of realizable applications is limited.

OAS as defined in the previous section aim at implementing smarter functionalities that can only be realized by using more extensive (environmental) knowledge, which can only be generated by arrays of diverse systems building a collaborative intelligence. In order to establish the access to and exchange of collaborative knowledge, closed systems need to be opened by introducing well-defined communication interfaces. In particular wireless communication with the outside world is new for embedded systems and thus demands the development of communication protocol stacks being able to guarantee quality of service during the collaboration of OAS. In the automotive domain, there exist ongoing standardization efforts for the realization of so-called *Dedicated Short Range Communication (DSRC)* among vehicles and infrastructure (Car2X) within the European [9, 10] and

North-American [11, 12] areas. An interesting aspect is that both standards explicitly address safety applications.

The possibility that OAS can have different adaptation configurations leads to challenges, when a collaboration among OAS should be planned top-down. In order to guarantee qualities like safety or functional performance, it would be necessary to test all combinations of the configurations of each participating OAS with respect to a specific collaboration scenario. This approach leads to a combinatorial proliferation and is thus not economical. Moreover, assuming that the different OAS are developed by different companies in isolation, it is very probable that no extensive knowledge is available at design time about systems and their possible configurations that the OAS under development should collaborate with.

In order to overcome this problem, a first step towards (self-)adaptation is to pre-engineer a set of variants for an OAS at development time and to realize the transition between these variants at runtime based on rules that assess the given runtime context. In this way, the single OAS are considered as black boxes whose potential diversities are constrained by explicit variants. More elaborated mechanisms for switching between variants are goal-based and take into account optimization potentials with respect to specific collaboration goals.

In conclusion, both openness and adaptivity aspects represent a challenge with regard to the prediction and assurance of collaboration qualities (safety, security, performance, etc.) for which no commonly accepted solutions exist yet. In order to handle the high complexity of OAS and to allow making reasonable predictions on their qualities, the following development aspects are of crucial importance (the driver is annotated in brackets):

1. Decoupling of the collaborating systems by the application of modularization principles (Openness)
2. Standardization and formalization of the communication interfaces between OAS. This also includes an explicit consideration of quality aspects (Openness)
3. Pre-engineering of system and collaboration variants (Adaptivity)

2.1.3 Service architectures

In the information system domain, the shift from component-based architectures towards service-oriented architectures (SOA) has been observable during the last decade. SOA is an approach to realize application functionality by the orchestration of highly decoupled black-box components that have well-defined interfaces (called service interfaces). SOA provides the following advantages:

Loose coupling The specific realization of a service is decoupled from its usage in shape of a formal service interface. Typically, service consumer components are also deployed on different physical entities than the service providers. Both of these aspects yield highly decoupled systems.

Standardization Due to standardized communication interfaces between service provider and service consumer, the infrastructure for service discovery, service announcement and the communication layer protocols can be separated from the service application logic.

Flexibility Since services are designed in a modular way, new system functionalities can be generated very flexibly by orchestrating different existing services.

Reusability Services typically provide access to small and cohesive tasks and thus allow an easy reuse in different applications.

Due to these advantages and the inherent distributed nature of OAS, service-orientation has been discovered as a suitable concept for modeling the interface between OAS, too. However, within the context of OAS, the quality of service with respect to quality attributes like safety, (functional) performance, timing, etc. is of essential importance. These qualities are typically not focused on by traditional SOA approaches and thus, they can hardly be applied as-is for OAS.

Within the automotive domain, there have been various attempts for the application of SOA: For example, [13] used a service-oriented approach for the improvement of traffic routing by making use of Car2Car communication. The distribution of sensor fusion across different cars in a cooperative adaptive cruise control scenario (CACC) implemented with SOA has been presented in [14]. One fundamental assumption in this case was that only the environmental situation around the collaboration changed, while the internal systems of the cars stayed static over collaboration time, which is typically not the case for OAS. Wagner et al. explicitly take the adaptivity aspect into account and proposed a SOA-based middleware solution as well as a development process guiding the creation of service architectures of distributed driver assistance systems in [15]. The approach is derived from IBM's "Service-Oriented Modeling and Architecture" methodology and is called *Service-Oriented Driver Assistance* (SODA). For the development and documentation of service models, they used the *Service Oriented Modeling Language* (SoaML) [16] which has been standardized by the *Object Management Group* (OMG) and is available as a UML profile. As part of the Conserts approach (see *Section 2.2*), a formalization of services in terms of both functional and quality dimensions, especially safety, has been proposed, too.

2.2 Conditional safety certification (ConSerts)

Having provided a basic understanding of OAS in the previous section, this section will focus on *conditional safety certificates* (ConSerts) [5], being the first methodological approach that tackles the openness and adaptivity challenges of OAS from the safety perspective.

2.2.1 Overview

The fact that OAS are often safety-critical systems, requires them to be certified before being released into operation. However, traditional safety certification relies on a complete system knowledge at design time, which is not given for OAS due to the reasons described in *Section 2.1.2*. Thus, safety engineering activities need to be shifted to runtime, where all design time uncertainties can be resolved. The idea of ConSerts for the solution to this problem is to follow a modular certification approach, which creates predefined modular safety certificates for (sub-)systems whose structures and behaviors are completely known at design time. These certificates are *conditional* in that they enable the support for several adaptation variants (=configurations) of a certified (sub-)system.

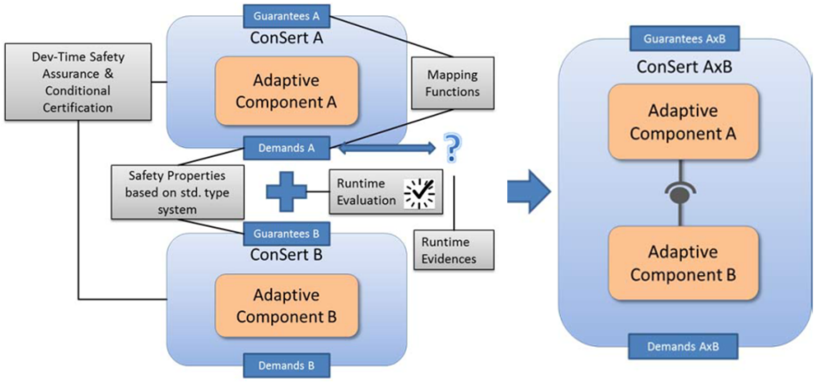


Figure 2.1: Conceptual overview of ConSerts (adapted based on [5])

Figure 2.1 shows the conceptual overview of ConSerts. ConSerts define safety contracts in terms of mapping functions that relate safety guarantees of an OAS to safety demands that are required to be fulfilled by other OAS or the environment. At design time, traditional safety assurance activities are carried out to assure that specific guarantees can be given for an OAS under the condition that associated safety demands are fulfilled. Adaptivity is supported by providing different guarantees varying with respect to the strength of the guaranteed properties meaning that weaker guarantees typically lead to weaker required demands. Multiple safety guarantee/demand relations with varying strengths are necessary to

increase the probability for a successful integration. The reason for this is the lack of a central integration responsibility, since this responsibility is dynamically distributed among the different OAS. As soon as several OAS are about to be integrated into a collaboration at runtime, their ConSerts are composed dynamically and thus, the collaboration's safety can be assured by checking if the demands of all collaborating OAS are fulfilled. This does not only include safety demands satisfied by other systems, but also properties that need to be satisfied by the collaboration environment itself, e.g. that the mandatory communication infrastructure is existent and properly initiated. Such demands are called *runtime evidences*. Eventually, to enable the dynamic evaluation of ConSerts at runtime, a machine-readable representation of ConSerts as well as technical solutions for their composition and analysis need to be available.

2.2.2 Operationalization of ConSerts

In order to operationalize ConSerts for OAS, an essential precondition is that the used architectural modeling approach supports the abstraction from concrete system interfaces in terms of a formal description of both functional and non-functional properties the systems provide to their environment. As elaborated in *Section 2.1.3*, service-oriented architectures support the required concepts in a powerful and efficient way by establishing provided and required services for systems that represent well-defined interfaces for provided and required black-box functionalities. Although the application of ConSerts is in general not limited to service-oriented architectures, the ConSert approach assumes systems to be architecturally decomposed with services due to the advantages of SOA.

Figure 2.2 depicts the ConSerts meta-model describing the relations between the elements used for the architectural composition of OAS by means of services (green area), the elements used for formalizing the safety properties of these services (red area) and the elements for the composition of safety contracts for OAS in shape of ConSerts (blue area).

For the application of ConSerts, it is sufficient to describe the functionality of an *open adaptive system* in terms of a set of *configurations*, which provide certain functional services to the environment and require other functional services to deliver their provided functionality. Configurations express pre-defined functional variants of OAS that result from their ability to adapt themselves to different runtime contexts mainly due to dynamic service availability of other collaborating OAS. A provided or required *service* as such first and foremost describes a functional purpose. This could be for example the provision of a speed value or an interface for receiving remote control commands from other systems. *Functional Service Types* serve in this respect to capture the detailed and formalized description of the services' functional aspects, which include properties like data types (e.g. numeric or compositional), units, valid value ranges, value resolution or provision frequency (continuous or event-based). In this way, each

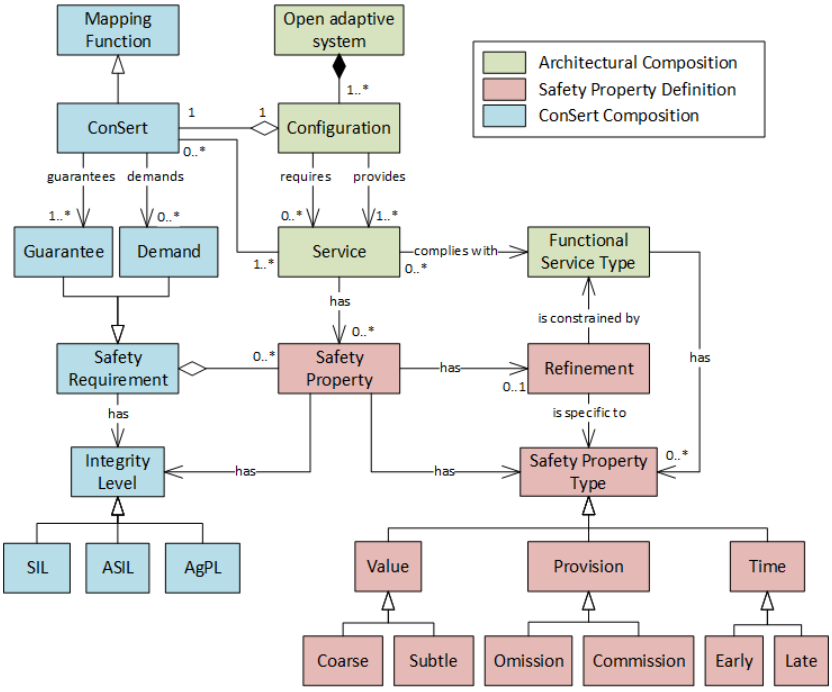


Figure 2.2: ConSerts meta-model

concretely instantiated *service* complies with exactly one functional service type. Note that the explicit separation of functional service types from their usage in concrete services allows to build domain-specific repositories of reusable service types. In addition to so-called *basic* functional service types, which represent the building blocks for the communication among OAS, there exists also a special service type referred to as *application service*. Rather than representing an interface between OAS, the application service provides the overall collaboration's functionality to human-beings and thus generates the desired business value of the collaboration.

In *Section 2.1.2*, the lack of consideration of quality properties within existing SOA approaches has been stated. The ConSert approach tackles this issue with respect to safety by enriching functional service types with safety-related information. In this way, a concrete service does not only comply with a functional service type anymore, but has associated *safety properties* that themselves comply with a *safety property type*. Safety property types describe possible failure modes of functional services or more precisely, they describe deviations from a specified functional behavior. *Figure 2.2* shows a commonly used classification for safety property types that distinguishes between value, provision and timing failures. In the same way like functional service types, safety property types could be organized in a domain-specific type system and assigned to specific safety properties. So far, a concrete service consists of its functional behavior

and safety properties that define possibly occurring failure modes for the functional behavior. With respect to the speed provision service example, one possible safety property type is *Value too high*. However, this information as such is not sufficient for describing a service failure in its entirety, because whether a too high speed value is critical for a collaboration may differ from scenario to scenario. Thus, when instantiating a safety property from a safety property type, it needs to be refined with respect to a specific collaboration scenario. This *refinement* has to specify quantitatively, when a certain deviation from the functional specification is considered to be a service failure. In addition, it can contain information on the possible consequences of the failure within the specific scenario. The knowledge on potential consequences of service failures is a precondition for assessing the risk of behavioral deviations expressed as refined safety properties. In summary, refined safety properties offer the possibility to specify *guarantees* or *demands* that assure with a specific level of confidence that certain *safety requirements* will be satisfied for a specific service. Put differently, the safety requirements assure with a certain confidence that behavior deviations of services do not exceed the specified boundaries. From a safety point of view, the collaboration partner providing the application service also has the special responsibility of providing the safety guarantees associated to the application service. These guarantees are direct translations of the collaboration's safety goals.

Having defined concrete services and their associated safety properties, the final step is the composition of *ConSerts* for the existing configurations of the OAS through mapping functions. These mapping functions relate guaranteed safety properties of provided services to demanded safety properties of required services or runtime evidences. By using Boolean functions for the mappings, dependencies between guarantees and demands can be expressed by common OR and AND relations. For each provided guarantee of each provided service, there shall be a separate ConSert Tree (CST), represented by a Boolean function $f : B^k \rightarrow B$. Inputs of the mapping function are k Boolean variables, each representing a demanded set of safety properties belonging to a required service. Such a Boolean variable is true if the demanded safety properties are actually met at runtime. Thus, if all Boolean variables that are logically related to a specific guarantee render true, the safety properties of that guarantee hold for the provided service. In a nutshell, ConSerts consist of multiple CSTs, which model the conditions for the guarantee variants of each provided service. *Figure 2.3* shows an example visual representation of a ConSert from the agricultural domain, where the described CSTs can be observed.

2.2.3 Validation of the ConSert approach

As yet, the ConSert approach has been evaluated in two case studies carried out in the agricultural and ambient assisted living domain.

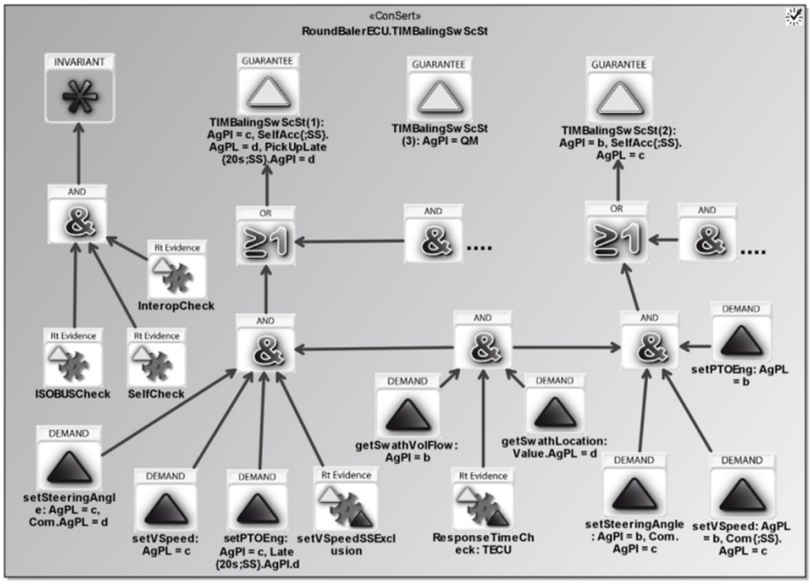


Figure 2.3: Visual representation of a ConSert (from [5])

In the first case, ConSerts have been applied to a tractor implement management system (TIM) for the following collaboration scenario: Tractors from manufacturer A should work together with round balers from manufacturer B. Since the round baler system has a better understanding on how concrete accelerations, speeds and steering should look like for creating perfect straw bales, the round baler should be able to control the tractor movement accordingly. In this way, the tractor provided a service to the round baler to control the tractor's speed. Besides the instantiation of ConSerts in the agricultural domain, the TIM case study also targeted the investigation of adequate engineering methods supporting the definition of ConSerts. The case study results showed that ConSerts can successfully handle TIM scenarios and that it proved useful to split the engineering activities into domain-level and system-level activities. This finding will be described in more detail in *Section 2.2.4*.

The second case study evaluated an emergency detection system (EDS) for the assistance of elderly people. Based on a set of already existent sensor and actuator devices for the health monitoring of the assisted person, ConSerts came into play for determining if sufficient safety guarantees can be given for the EDS, when new devices from a health monitoring device market place should be integrated into the EDS. The main characteristic of this scenario was the amount of different devices that were available, which led to the result that an adequate domain-level standardization of service types and safety property types needs to be carried out. Another result was a proof-of-concept of the implemented runtime evaluation procedures for ConSerts.

2.2.4 Engineering of ConSerts

The ConSert evaluation case studies have shown that it is useful to split the engineering activities enabling the composition of ConSerts in domain-level and system-level activities.

Domain-level engineering

The observation that collaboration scenarios within specific domains often use similar basic functional service types which also have similar safety property types, led to the idea to create a domain-specific repository for these elements, the so-called *Safety Domain Model* (SDM), which is shown in *Figure 2.4*. Although concrete OAS are not completely known at design time, basic functional service types together with their safety property types can be assumed to be required for several collaboration scenarios within a domain. To that end, the SDM captures this knowledge so that it can be easily reused in the system-level engineering phase, when the ConSerts for concrete OAS have to be created.

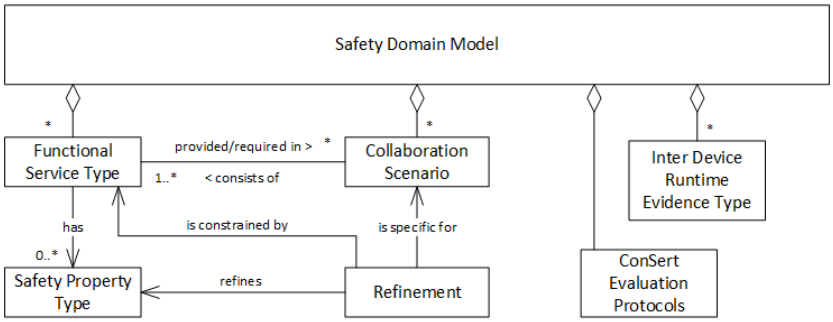


Figure 2.4: Safety domain model (adapted based on [5])

Collaboration scenarios can be thought of as typical interaction patterns that occur within a specific domain. Such interaction patterns include participant roles, structural information about the role interfaces in shape of functional service types as well as interaction schemes. Assuming that a comprehensive set of basic service type specifications exists for an application domain, safety property types can be derived by the application of suitable safety analysis techniques. [17] recommended deductive *hazard operability studies* (HAZOP) for the safety analysis of service types, since the causal model of HAZOP matches the idea that failures occurring in concrete OAS will eventually manifest themselves at the service-level and will therefore have consequences on a given collaboration. In addition to the identification of functional service types and safety property types, *inter-device runtime evidences* have to be standardized on the domain-level as well, since their evaluation involves the interaction between multiple participant devices/roles. Thus, the interaction protocols for this evaluation need to be specified on the domain-level. Furthermore, in particular

domain-specific communication protocol stacks have to be considered for the technical realization of ConSert composition and evaluation and thus ConSert evaluation protocols have to be standardized on the domain-level as well.

Despite the advantages provided through an existent SDM, there exist some challenges for its creation, too:

1. The creation of the SDM includes a high amount of preparation work, during which no direct benefit can be generated.
2. Domain engineering relies on the background knowledge of experts rather than on real systems and thus the standardized elements might not fit concrete required usage scenarios.
3. Standardization for an entire domain needs common agreements within a majority of companies of that domain. Such agreements can be negatively influenced by communication challenges and competition.
4. A suitable level of abstraction needs to be found for the SDM to leave enough space for solution flexibility in concrete OAS realizations.

However, having the identified challenges of domain-level engineering in mind, a diligently maintained SDM can be highly beneficial for the industry in the long run. The fact that technology research and innovation as well as experience are key enablers for new collaboration scenarios, also means that the SDM will be subject to a continuous evolution.

System-level engineering

Having the safety domain model definition as a basis, it can be used for the creation of ConSerts for concrete OAS that should later on participate in collaboration scenarios. *Figure 2.5* shows the relation between domain- and system-level activities through the SDM.

When a new OAS should be developed or extended to support a specific collaboration, the first step is to explore the safety domain model for available information on the desired collaboration scenario and its variants. The selection of variants that should be realized in the OAS under development directly guides the determination of the functional services that have to be provided by the OAS. Selecting several variants implies the realization of multiple configurations where each configuration describes exactly one supported collaboration variant.

Based on configurations describing provided and required services, safety goals and associated safety guarantees have to be determined, which can be created by means of traditional safety engineering techniques like *hazard and risk assessment* (HARA) techniques. Obviously, the selection of multiple collaboration variants leads to higher development costs due to the provision of higher safety guarantees. However, the business benefits that can be leveraged from a better collaboration performance in case of

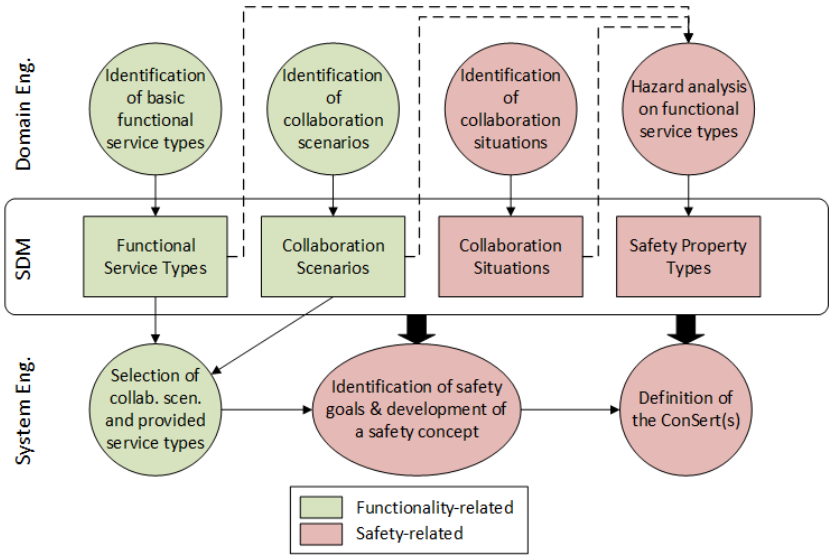


Figure 2.5: Engineering activities and the SDM (adapted based on [5])

higher provided guarantees, can make this investment attractive. Thus, the main engineering goal is to provide sufficient guarantees for the provided services of the collaboration variants which have been selected according to business decisions.

Once the safety goals and their associated safety guarantees have been derived, a safety concept has to be developed that provides an argumentation ensuring that the safety goals are satisfied. The creation of a safety concept starts with a safety analysis of the OAS under development which explores the causes that can lead to a violation of the safety goals. The safety property types defined in the SDM can act as a starting point for the system-level safety analysis. The safety goal violation causes can either be located internal in the OAS under development or can be caused externally by required services or runtime evidences. In any case, the safety concept should address the identified causes with appropriate countermeasures or additional safety demands. The formalization of the relation between safety guarantees and safety demands (=the ConSert(s)) of the OAS under development can be carried out based on the contents of the safety concept.

The final step is the actual certification of the OAS under development: After an extensive examination of the documentation of the safety goals, the safety concept and the derived ConSerts, an authorized certification body will issue the required safety certificate. The required documentation could be for instance organized in a safety case, while the actual structure of the safety case might be dictated by domain-specific safety standards.

	Vehicle type	Control	Traffic integration	Sensors	Goals
SARTRE	Mixed	Lat Long	Highway, Mixed	Production	Comfort, safety, congestion, energy saving
PATH	Cars or Heavy	Lat Long	Dedicated lane	Mixed	Increased traffic throughput, energy saving
GCDC	Mixed	Long	Mixed	Mixed	Acceleration cooperative driving system deployment
Energy-ITS	Heavy	Lat Long	Dedicated lane	SotA	Mitigate lack of skilled drivers
SCANIA	Heavy	Long	Highway, mixed	Production	Commercial fleet, energy saving
KONVOI	Heavy	Lat Long	Highway, mixed	Production	Traffic throughput, driver acceptance, legal implications
This thesis	Heavy	Long	Dedicated lane	Mixed	Safety, vehicle diversity, engineering process

Table 2.1: Comparison of existing platooning projects (extended based on [18])

2.3 Truck platooning

Since the thesis goals described in *Section 1.3* explicitly focus on truck platooning scenarios in the automotive domain, this section gives an overview of the state-of-the-art for the realization of truck platooning and similar scenarios. In particular, it will be explained, how safety has been addressed for truck platooning in the past.

2.3.1 Research projects

An overview of the five most notable research projects that have dealt with vehicle platooning systems in recent years is given in [18]. More specifically, the following projects have been considered: *Safe Road Trains for the Environment* (SARTRE) – a European platooning project, PATH – a California traffic automation program, *Grand Cooperative Driving Challenge* (GCDC) – a cooperative driving initiative, SCANIA platooning, the German KONVOI project (RWTH Aachen) and *Energy ITS* – a Japanese truck platooning project.

All of these projects have in common that they solve the platooning problem through the exchange of information through vehicle-to-vehicle com-

munication. However, the provided platooning concepts as well as the technical solutions and goals differ due to different research motivations within the projects. On a high level, the principal variations in the examined platooning scenarios are compared with respect to the following five parameters in *Table 2.1*.

Vehicle type Indicates whether a platoon can consist of cars, heavy trucks or a combination of both ("mixed")

Control Indicates whether the projects consider only the longitudinal control of the platoon or in addition the lateral control

Traffic integration Indicates whether other traffic participants (cars or trucks) not belonging to the platoon are considered. "Highway, mixed" represents a multi-laned highway where cars, trucks and the platoon are driving in a realistic setting. In contrast, a single dedicated lane assumes that there is no other traffic than the platoon itself.

Sensors Indicates whether sensors have been used that are already in production ("production"). "SotA" implies either that the sensors are currently infeasible or too expensive for commercial production. When a project is annotated with "mixed", both types of sensors have been used.

Goals The goals that guided research for the projects

Refer to [18] and its reference list for more technical details on the presented projects.

2.3.2 Truck platooning safety

Apart from different existing platooning concepts and technical solutions for their realization, it is in particular interesting in the context of this thesis, how safety has been addressed in platooning research. To that end, [19] investigated in a systematic literature review (SLR), what is currently known about safety for vehicle platooning including which analysis methods have been used, which hazards and failures have been identified and which solution elements have been proposed to improve platooning safety. In addition, a gap analysis has been provided identifying outstanding research questions for which no sufficient solutions exist yet. Note that the systematic literature review also covers safety-related research of all projects given in *Table 2.1*.

Although a majority of publications within the vehicle platooning area mention safety as an important aspect for the realization of platooning, only few of them give explicit proposals how to achieve safe platooning. An exception to this fact is the extensive study of *string stability*, which is a necessary prerequisite to avoid vehicle collisions within a platoon due to disturbances in the control system. Since string stability is mostly studied for the intended platooning collaboration (i.e. under the absence of

failures), its achievement does not give sufficient argumentation on the collaboration's safety.

With respect to safety analysis methods, the SLR has found that analyses have been conducted on two levels: First, on the system safety level, where the comprehension of the system's safety as a consequence of the integration of components, users and the environment has been analyzed. Second, the analyses of specific aspects such as component failure consequences or certain kinds of damages have been carried out.

Interestingly, only very few publications mention established system safety analysis methods like fault tree analysis (FTA), Failure Mode and Effects Analysis (FMEA) or Hazards and Operability Analysis (HAZOP). [20], who tried to apply the ISO 26262 to a platooning system, state in this respect that the standard is insufficient, when the safety analysis scope changes from a single vehicle to cooperative systems. More specifically, they deem the severity classification of hazards as an issue and therefore propose a fourth severity level also leading to an additional ASIL level E. The assessment of risks for vehicle collisions influenced by different vehicle spacings was subject of the studies in [21]. They analyzed how different vehicle spacings can lead to casualty risks, which allow a fatality probability estimation by applying the Abbreviated Injury Scale (AIS) to different collision speeds. There also exist theoretical approaches for the identification of safety regions that are based on control or game theories.

The picture on **hazardous situations** and **accidents** that occur in platooning scenarios is quite consistent in literature and can be summarized as:

1. A vehicle in the platoon runs into a preceding vehicle and this becomes an issue, when the first vehicle needs to perform an emergency braking maneuver.
2. A platoon vehicle crashes into another vehicle during cut-in situations, where the other vehicle changes lanes to end up in the middle of the platoon.
3. Harm is caused due to a platooning vehicle exiting the road in an attempt of trying to avoid a collision by steering.

With respect to longitudinal control of platooning vehicles, two **safety goals** are identified as a consequence of the actuation of both brakes and accelerator:

1. No unintended full braking in platooning at cruise speed
2. No sudden unintended full acceleration in platooning at low speed

A categorization for potential sources of technical (component) failures, which could guide safety analyses, include the areas sensors, actuators, inter-vehicle communication, computation, internal communication networks, driver interfaces and infrastructure. Especially for system collabora-

tion scenarios, interactions between several failures of these classes have to be considered, too.

For the general **handling of failures**, some strategies have been proposed that introduced a number of levels characterizing safe states ranging from the immediate stop of the vehicle as the most severe to no required special handling at all. There exists a general agreement on the basic technical solutions that affect platooning safety in a positive way. With respect to sensors, redundancy and sensor-fusion concepts have been identified as very important. The main focus regarding actuation is residing on redundant secondary brake systems or to implement two brake actuators. A clear conclusion among the research community is that platooning approaches without communication are insufficient. It is suggested to apply redundancy patterns as well for inter-vehicle communication, e.g. to complement radio communication with infrared or visible light communication.

Another relevant aspect for platooning safety is how differences in vehicle configurations affect safety and how safe platooning can be realized in presence of these variabilities. However, the majority of reviewed studies assume that vehicles participating in a platoon have similar capabilities, especially with respect to braking, which is judged as the most safety-critical behavior during platooning. The only known solutions to this issue reside on the management level and advise either to order the vehicles according to brake capability with the least capable being the platoon leader or to limit the brake force of the leader to that of the least capable vehicle. A general problem with respect to vehicle variabilities is the lack of models to provide reliable data on vehicle capabilities such as available brake forces and the lack of understanding what data accuracy is required for these capabilities.

In summary, it can be stated that the realization of safe vehicle platooning clearly lacks of systematic safety analysis using well-established techniques. In addition, standards like ISO 26262 or IEC 61508 have been rarely explicitly addressed in the context of vehicle platooning. A possible explanation could be that these standards are insufficient to deal with cooperative systems in their current state. From an economical point of view, vehicle platooning will only be a widespread success, if diversities in vehicle characteristics are explicitly considered and the formation of platoons is not restricted to vehicles from the same brand. Thus, new actors like standardization bodies come into play that have to define interoperability requirements for the interactions between vehicles of different brands.

Although not focused on the automotive domain explicitly, *RailCabs* (Uni Paderborn) [22] is nevertheless an interesting research project, because it deals with the platooning problem for self-adaptive cyber-physical systems in the railway domain. RailCabs are autonomous railway vehicles that may form platoons automatically on track without mechanical coupling. The performance requirements of the RailCab system are higher than those for

typical vehicle platoons, since RailCabs drive at 160 km/h with very small distances of less than one meter. In order to satisfy these requirements, the platoon leading RailCab functions as a coordinator who is responsible for deciding, when and where other RailCabs may safely join or leave a platoon. In addition, the coordinator announces all acceleration and braking maneuvers before they are started, since feedback controllers cannot react fast enough to guarantee safe and stable platoons. This is a major difference to conventional vehicle platooning, because there the leading vehicles cannot announce maneuvers *before* they occur, since these are driven by human-beings whose behavior cannot be predicted. The strategy for guaranteeing safety in the RailCabs system relies on formal verification techniques based on assume-guarantee reasoning.

3 Solution overview

This chapter first introduces the truck platooning scenario in detail as it will be used throughout the case study conducted in this thesis. Afterwards, the challenges that needed to be solved for achieving the thesis goals (see *Section 1.3*) will be put into a big picture and exemplified by means of the running example. It will be stated to which extent existing methods can address the challenges and where new solutions had to be developed within this thesis. Thus, this chapter serves as a high-level road map through the contents of this thesis.

3.1 Running example

Concerning the choice of a suitable application scenario for which the case study of this thesis should be conducted, there have been various candidates in the context of OAS to choose from. For the reasons stated in *Section 1.1*, the automotive domain and more specifically a truck platooning scenario have been chosen. Various research groups and industry companies have already studied truck platooning from different research angles, mostly embedded in internationally funded research projects. Details on these projects as well as an overview on their efforts with respect to safety assurance are given in *Section 2.3*. The research budgets invested in truck platooning in general show that the mastering of truck platooning scenarios is of high practical relevance.

This thesis' case study is aligned with the *European Truck Platooning Challenge 2016* (ETPC) [23], an international research project initiated to boost the evolution of truck platooning within the European Union. The focus area of this challenge is that two or more trucks of *different vendors* can build platoons dynamically. This includes a step-wise increase of the level of automation so that the necessity of having human drivers on every truck can be finally shifted to having only one driver operating the leading vehicle of the platoon. In this way, the transport efficiency is augmented by eliminating the need to for follower drivers to be subject to stipulated driving time and rest period limitations.

Apart from enabling business benefits, another major goal of the ETPC 2016 is the improvement of traffic safety. The fact that truck platooning scenarios are realized by OAS and are inherently safety-critical, i.e. all safety life cycles activities need to be performed, shows their suitability for an evaluation of the ConSert approach on a full scale.

3.1.1 Truck platooning scenario description

In order to identify the different functional aspects inherent to truck platooning, a typical sequence of its steps is described in the following realistic scenario:

As a mission goal has been defined that two heavy-duty transportation trucks with up to 40 tons of weight have to transport their goods from Stuttgart, Germany to Rotterdam, Netherlands [24]. The route only consists of well-developed multiple lane highways each direction. Having entered the first highway section in Stuttgart, the human drivers are still driving their trucks manually. Next, both get an indication on their dashboard displays that there are trucks in the reachable area supporting the building of a platoon. After both drivers actively opt in, the platooning system determines based on vehicle characteristics, which of the trucks should lead the platoon and finally the follower driver hands over the driving control to the system. At that point, the inter-truck distance is still 50 meters which is the minimum legally allowed distance between trucks on German highways at speeds higher than 50km/h [25]. Since the platooning system has a much faster reaction time than a human-being, it slowly decreases the distance from now on until a value is reached so that the worst-case maneuver, namely a full stop of both trucks with maximum possible braking forces, will not lead to a frontal crash. This so-called safe distance may vary based on different runtime context influences:

- system-external influences such as weather or road surface conditions
- system-internal influences such as the current truck speeds, tire wear, brake capabilities or truck weight

The platooning system continuously receives these runtime conditions for all platoon participants through wireless communication among the trucks and reacts to condition changes by either increasing or decreasing the distance. It also detects lane change maneuvers of the leading truck and takes care that these operations are safely performed for both trucks and all other road users. Finally, when the platoon is approaching the highway exit at Rotterdam the platooning system should notify both drivers that according to their mission goals the platoon is going to be dissolved soon and thus the follower driver needs to prepare for taking over manual control again. After the notification has been triggered, the system increases the distance to 50 meters again, where the follower driver resumes manual control by pressing a dedicated button on his user interface.

3.1.2 Scenario constraints

Figure 3.1 shows the different functional blocks extracted from the scenario description above. The three main functions are *Platoon building*, *Platoon driving* and *Platoon dissolving* and have been further broken down into more explicit functionalities. Note that some connections as well as

irrelevant refinements of functions have been omitted due to diagram simplicity.

Both *Platoon building* and *Platoon dissolving* include switching the truck control between system and human through the communication between drivers and their user interfaces. In the context of autonomous driving, the aspects of these functions are focused on by whole research communities like cognitive sciences or human computer interaction [26] and should for the sake of simplicity be excluded from the scope of this thesis. The same holds for *Platoon driving*'s sub functions *Lateral control* and *Monitor other road users*. All excluded functions and actors are colored in grey in *Figure 3.1*.

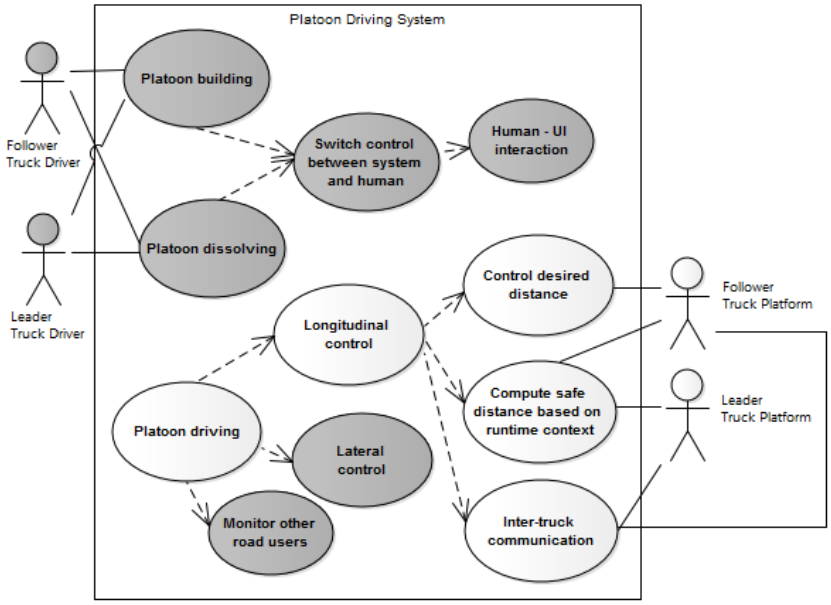


Figure 3.1: Truck platooning high-level functional net (grey = out of this thesis' scope)

Recalling the purpose of the case study, namely to apply the ConSert approach for yielding a safe collaboration during *Platoon driving*, the *Longitudinal control* function contains the relevant conceptual parts to demonstrate all aspects of the approach. As soon as an application guideline has been derived based on the *Longitudinal control*, it should be easily applicable for similar functions like the *Lateral control* as well without further ado.

The exclusion of *Lateral control* implies that in the running example there is only one lane on a perfectly straight road. Furthermore, no other traffic such as cars, trucks or any other obstacles are existent. With respect to the earlier mentioned runtime variabilities, the following ones shall be explicitly considered:

– Truck weight including the current cargo weight
– Maximum brake capabilities of the brake systems
– Road inclination
– Road surface conditions (Icy, Rainy, Dry)
– Weather conditions having an impact on the accuracy of visual sensors

3.2 Solution big picture

This section will put the thesis goals as set out in *Section 1.3* into relation and shall thus serve as a high-level overview into which the specific activities of the construction of both safe nominal and safe fail behavior specifications will be embedded in *Sections 3.2.1* and *3.2.2*, respectively.

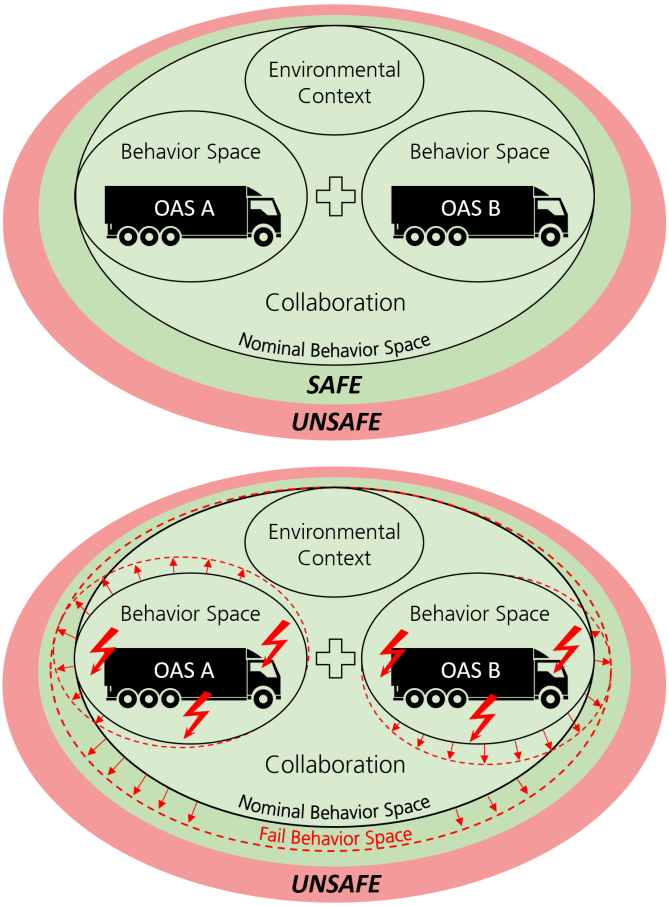


Figure 3.2: Nominal behavior vs. fail behavior of OAS collaborations

Figure 3.2 depicts the relation between safe nominal and safe fail behaviors in the context of OAS. In order to enable a specific collaboration behavior between different collaboration partners being an OAS each, the OASs themselves have to provide certain behaviors through their functionalities. A set of behaviors that is observable for a certain OAS during a collaboration is depicted as a *Behavior Space* in the figure. Given a specific *Environmental Context* during runtime, the *Nominal Behavior Space* of the overall collaboration is spanned through the combination of the environmental context and the behavior spaces of the single OASs. The *Nominal Behavior Space* explicitly only includes behaviors that arise from the intended functionality of the collaboration. Obviously, all observable collaboration behaviors should be safe at any time and thus the nominal behavior space should be completely subsumed by the space of safe behaviors (depicted as the green area in the figure). Unsafe behaviors that potentially lead to accidents are represented with the red area in the figure. Summing up the upper part of *Figure 3.2*, the behaviors of the collaborating systems need to be specified in such a way that the collaboration's nominal behavior will be safe in any given environmental context.

Like closed systems, open adaptive systems are prone to the occurrence of random hardware and systematic software failures, too. Under the influence of failures, the observable behaviors of the collaborating OASs deviate from their specified intended functionalities. Thus, the behavior space for each OAS is extended with behaviors resulting from deviations from their intended behavior (indicated by red arrows in the lower part of *Figure 3.2*). The extensions of the single OASs' behavior spaces lead to an extension of the overall collaboration's behavior space as well, which is denoted in this thesis as *Fail Behavior Space* and indicated with a red dashed ellipse in the figure's lower part. Note that the term fail behavior can only be used when talking about the overall collaboration, because whether a certain behavior is a fail behavior (referred to as *behavioral malfunction* in ISO 26262 [27]) or not cannot be decided for a single OAS without considering the collaboration context. Thus, an OAS can only provide a set of observable behaviors, which *might* be considered hazardous within specific collaborations while in others, they are not safety-critical at all. Consequently, instead of dealing with fail behaviors for single OASs, we can only observe deviations from their nominal behaviors [28]. If these deviations exceed certain limits, the resulting *Fail Behavior Space* of the overall collaboration is extended in such a way that it might overlap with the space of unsafe behaviors (red area). Therefore, in order to guarantee a safe collaboration also during the occurrence of deviations from the nominal behavior, the deviations need to be explicitly bound by design in such a way that also the fail behavior space is completely encompassed by the safe behavior space (green area).

Considering the running example, a nominal behavior of the follower truck could be *Braking with maximum brake force*. Due to a systematic software failure within the follower truck, the effectively applicable brake

force is reduced to 80% of the originally assumed maximum brake force. Thus, the brake force is deviating from its nominal value by 20%, more specifically, it is 20% lower than assumed. Whether this deviation is dangerous for the given platoon (i.e. the collaboration behavior is unsafe and leads to a crash), it can be explicitly controlled by the distance that is maintained between the trucks. If the distance is high enough, the trucks will not crash even if the actual follower braking capability is lower than assumed. In the reverse conclusion, the more conservative the separation is chosen, the higher the degree of deviation is tolerable without risking unsafe platooning. However, a more conservative distance will lead inevitably to a worse platooning performance in terms of fuel saving. Apart from the fact that deviation bounds need to be explicitly quantified, this example shows that a trade-off has to be found between collaboration performance and behavior deviation tolerance.

Concluding from the above thoughts, the first necessary activity for achieving a safe collaboration is the specification of a safe nominal behavior for the collaboration, which includes the derivation of nominal behaviors for the collaborating OASs as well. Afterwards, the potential deviations from the nominal behavior need to be analyzed and reasonable bounds have to be explicitly defined quantitatively for each deviation. The definition of these bounds has to be carried out in such a way that firstly, the collaboration stays safe although a deviation occurs and secondly, the trade-off between collaboration performance and failure tolerance is resolved optimally. In this way, the main constituents of the safe fail behavior specification will be the characterization of behavior deviations and their quantification. In the following subsections, the sequence of activities for the construction of both safe nominal and safe fail behavior specifications will be elaborated in more detail.

3.2.1 Safe nominal behavior specification

Having explained the terminology and most important concepts of the development of OAS in *Section 2.1*, this section will exemplify the challenges of the safe nominal behavior specification (Thesis goal 1) by using the running example. To this end, *Figure 3.3* shows an overview of the activities (boxes) as well as the main input and output artifacts (circles).

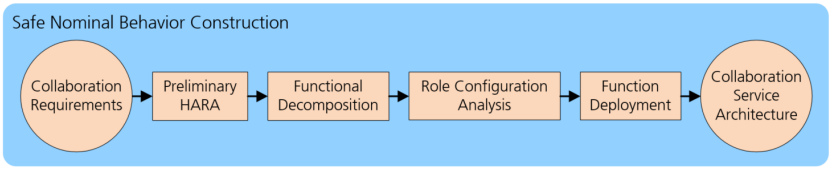


Figure 3.3: Safe nominal behavior specification – Activities overview

The main input for the safe nominal behavior derivation is a set of requirements for the intended collaboration. The key functional **requirements** of the running example have been described in terms of a scenario and associated use cases in the previous section. Based on these requirements the functionality needs to be decomposed hierarchically.

The first challenge in this respect has been to find a starting point for the functional decomposition. Having the goal in mind to construct a *safe* nominal behavior, hazards caused by the intended functionality need to be avoided from the outset. However, functional safety standards like ISO 26262 do not address this so-called safety-in-use yet [29], although there are ongoing standardization efforts with respect to the *safety of the intended functionality*[1]. In the context of OAS, the safety of the intended functionality is more critical than in closed systems, because OASs will be only integrated at runtime and thus a guaranteed safety-in-use is questionable due to the degree of uncertainty about the integration context at development time. Thus, the accidents that can occur through the intended collaboration have to be known explicitly before any functional design can materialize. Therefore a **preliminary hazard and risk analysis (HARA)** has to be carried out so that the safety goals that should be met by the nominal behavior are known. Due to the constraints of the running example (single lane, no lateral motion, no other road users but two trucks), there is only a limited number of critical accidents that can occur, namely a frontal crash, where the follower truck crashes into the leader truck. In order to find the hazardous events leading to an accident, existing HARA approaches examine malfunctioning behavior in certain operating modes and operational situations. In the context of OAS, this approach yields problems, because whether a certain behavior like "Leader is braking" is malfunctioning or not can hardly be judged without additionally considering the state of the collaboration scenario, namely if the distance between follower and leader is large enough given the current runtime context. Trying to anticipate the effect of a behavior in all possible and potentially unknown collaboration situations will likely lead to a combinatorial explosion. In order to avoid this problem, we formalized the accidents in terms of physics and formulated their negation as *safe conditions*, which form the basis for the functional decomposition to make sure accidents are avoided by design. In terms of truck platooning the safe conditions pose constraints on the motion state of the collaboration, namely that the distance between the trucks must not fall below a certain value.

The distance between the trucks is thus a physical entity that has to be controlled by the platooning function, i.e. it is a functional output representing a set point. However, the specific value for the distance being considered as safe is dependent on the characteristics of the trucks and the environment at runtime, which are inputs to the platooning function. The **functional decomposition** can be carried out by initially creating

[1] `http://www.iso.org/iso/catalogue_detail.htm?csnumber=70939`

one function per safe condition that takes care that the condition will be met. The inputs of these functions will be hierarchically decomposed by using physical laws until only physical entities are left that can be directly measured or determined. The function outputs directly refer to set points, for which other physical entities need to be found that are able to realize these set points. This strategy of decomposing the system function in two different directions (towards inputs and outputs) is called *horizontal refinement* (also known as superposition refinement) and is described [30]. The eventual goal of the functional decomposition is to have a mathematical relation between the measurable inputs and actuatable outputs of the platooning function.

Having a functional model of platooning in place, the next step in traditional system development would be to refine the functional model into a more technical one considering the details of *both* actual truck platforms. However, this approach assumes these details are known in their entirety, which is not the case for OAS. This can be observed in the running example, where we have two trucks that should both realize truck platooning, but are developed by two different manufacturers in isolation. Thus, a big challenge is to deliver as much constraints to each manufacturer as necessary for implementing platooning in a safe manner, but still leaving enough space for design flexibility. To that end, instead of deploying functions directly to concrete truck system components, we deploy functions to an additional layer of abstraction, which is comprised of the roles of a collaboration. Within the running example, these roles are leader and follower. This abstraction is capable of sufficiently describing the collaboration interface, which is essential in OAS and at the same time allowing one concrete truck to implement only leader, only follower or even both roles. Furthermore, the role abstraction helps keeping the development focus on the platooning domain instead of drifting into technical details too much.

Considering the overall platooning function, all input values that need to be measured and all output values that will be actuated are known at this point. In order to be able to deploy the platooning sub functions to roles in an optimal way, knowledge about the probable source of input values is needed, e.g. the speed value of the leader role will likely be originating from the leader truck eventually. In a **role and configuration analysis**, assumptions have to be made about source and sink roles of input and output values. This information can be generated by looking at existing state-of-the-practice/state-of-the-art sensors and actuators. Since nowadays, there exist a lot of truck variants with highly varying hardware configurations, one challenge faced in this thesis was thus to pre-engineer a decent set of role hardware configurations that can be found in today's trucks to enable the platooning scenario for as many trucks as possible.

As soon as the origin roles of functional input values are determined, the functions can be deployed to roles. The possibility of communication between leader and follower allows for great flexibility in deployment, be-

cause a value can be e.g. measured at the leader but used at the follower after communication happened. Thus, strategies needed to be developed that allow an optimal **function deployment** with respect to certain criteria like communication medium occupation and more important that the deployment can be carried out systematically. This was solved by starting at the platooning function inputs and traversing along the signal propagation paths, while deciding for each function based on a set of heuristics to which role they have to be deployed.

Having deployed functions to roles, the collaboration interface is sufficiently defined, i.e. which information has to be communicated how often from which source role to which target role. Every information that is passing the collaboration interface in this way can be formalized in terms of services as defined in *Section 2.1.3*. The communication direction indicates, which role needs to provide or require a certain service. The service derivation has to be carried out for all supported collaboration configurations. As described in *Section 2.2*, the definition of provided and required services between roles is not sufficient in order to ensure a safe overall collaboration. Even if all required services of all roles are matched by provided services of other roles, this only ensures the safe nominal behavior for each role, but not for the whole collaboration itself. Therefore, it is required to have one function that is responsible for providing the application service, namely a safe platooning collaboration, to the truck drivers. This means this function has to guarantee that all safe conditions are met during truck platooning. Finding a reasonable choice for the assignment of the application service has been another challenge for the safe nominal behavior specification. Having service and function descriptions as well as their deployments to roles documented in terms of a **collaboration service architecture**, truck manufacturers could in principle take this documentation and implement a certain role for an existing or new truck according to classical system development methods. To this end, the truck manufacturer only has to adhere to the service interface that has been assigned to the role(s) to be implemented. Note that a modernly equipped truck could in principle implement multiple configurations of a role at the same time, which increases chances that it will eventually find another truck at integration time (=on the highway) matching one of its service interfaces.

In summary, a safe nominal behavior specification in the shape of a collaboration service architecture enables the implementation of different collaboration roles in isolation, while still guaranteeing a safe overall collaboration at the runtime integration. However, the safe nominal behavior specification does not explicitly take into account possible random hardware or systematic software failures yet. Thus, additional activities have to be carried out to guarantee a safe collaboration also in the presence of service failures, which are addressed by ConSerts and exemplified in the next section.

3.2.2 Safe fail behavior specification

Based on a safe nominal behavior specification, this section exemplifies the challenges that appeared during the activities in order to construct a safe fail behavior for the truck platooning scenario (Thesis goal 2). *Figure 3.4* shows the activities that need to be carried out to produce a formal ConSert model having a properly defined service architecture of the intended collaboration as input product. The terminology and concepts of the ConSert approach that will be used in this section have been introduced in detail in *Section 2.2*.

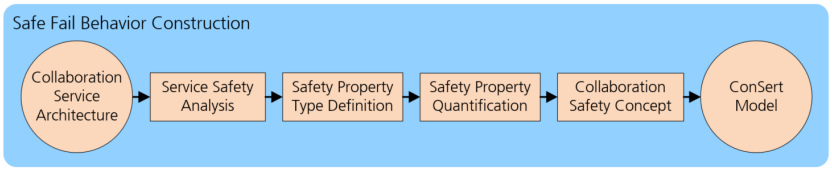


Figure 3.4: Safe fail behavior specification – Activities overview

The given collaboration service architecture contains a hierarchical structure of services where the top-most element is the application service that is provided to human-being(s) and has been allocated to be provided by a specific collaboration role. In order to assure that the application service can be delivered safely certain services have to be provided from collaboration roles correctly that might themselves demand correct service delivery from other roles. The first challenge in this respect has been to analyze which types of service failures may occur for any given service so that the application service cannot be provided properly anymore. The related activity for achieving this goal is the **service safety analysis**. Note that this analysis focuses only on service failures occurring at the collaboration interface between roles and does not try to identify the causes for these failures within the internal functional specification of the roles. In terms of truck platooning, consider the follower role is requiring a service from the leader role containing a continuous provision of the leader's current speed. If the follower role receives a wrong speed value for different reasons be it timing, value or provision failure, it can only react to that value as it is perceiving it and thus it is irrelevant from this black box perspective, how the erroneous speed value was caused within the system implementing the leader role.

While the service safety analysis is focusing on the activity of systematically collecting possible service failures, the main challenge of the **safety property type definition** is to enrich the functional service descriptions with their possible service failures in a more extensive way. This includes the documentation of the potential deviations from correct service as well as their effects on the concrete collaboration. Considering the leader speed provision service example, a "value too low" failure could be characterized with the deviation "The provided speed value is

more than x $\frac{m}{s}$ lower than the actual leader truck's speed" which could potentially have the effect that "The maintained distance between leader and follower truck is unsafe".

Since the system under development should eventually prevent the effects specified within the safety property types from happening, the negation of the safety property types directly represents safety requirements the system has to fulfill. From the perspective of a role using a service that means that it can only react safely to a failure, if the bounds of the service failure's potential deviation from correct service are explicitly quantified. This matches the common understanding that well-written requirements should always be testable, which is highly supported by a quantification of the requirement. Based on this **safety property quantification**, fault tolerance mechanisms can be eventually implemented. The determination of *reasonable* bounds for deviations has been a tough challenge in the course of this thesis, because the effects of one deviation or even a combination of multiple simultaneous deviations on the collaboration's safety are hard to judge without concrete evidence in today's complex systems. During platooning for instance, it is not directly clear, how much additional distance is needed between both trucks to compensate for a leader speed value being $5\frac{m}{s}$ too low. The fact that the effects need to be known for complete ranges of deviations, makes the situation even more complex. Thus, in order to come up with reasonable bounds for the specification of service safety properties, a simulative approach has been carried out in this thesis. To that end, a Matlab/Simulink model has been created for the platooning scenario that enables the failure injection at the service level. Based on batch simulations of different service failure combinations, an informed decision can be taken with respect to the trade-off between tolerated deviation ranges and their cost in terms of performance losses. With respect to platooning, this means that by tolerating a higher speed value deviation, the additional distance that is needed between the trucks for still maintaining a safe collaboration, is a performance loss, because a higher amount of fuel is needed for the follower truck.

Having augmented the services with quantified safety properties, a **collaboration safety concept** has to be created that includes a sound reasoning for the fulfillment of the top-most safety goal, namely that a frontal crash between follower and leader truck must be prevented during platooning. The main challenge within this thesis that arose from this task was to find a structure for the safety concept that can be systematically derived from existing artifacts and that can be reused for similar service-based OAS. Starting at the application service, the service hierarchy has to be traversed top-down. On the one hand, an argument needs to be provided for each role that the safety properties of its provided services are guaranteed under the assumption that its safety demands towards other roles are fulfilled. On the other hand, the collaboration safety concept needs to assure that if all safety demands of all roles are fulfilled, the application service can be guaranteed in a sufficiently safe way as well.

In summary, each role provides certain services within the collaboration that have to guarantee quantified safety properties. In order to provide these services with the desired quality, the role may also require services from other roles while demanding specific bounds for the maximum deviation of correct service. This information is finally compiled into **ConSert models**, which contain a machine-readable representation of both demands and guarantees for a specific role within a collaboration. Within the truck platooning scenario it can be assumed that both trucks implement the follower and the leader role. Based on the concrete configurations of the trucks, there exist thus at least two associated ConSerts for each truck, describing the provided/required services when acting as leader or follower. Prior to building a platoon on the highway, both trucks evaluate their ConSerts hierarchically starting in the application service and if all required services of both roles find a matching service provision with sufficient quality, a safe platooning collaboration can be assured.

Since a ConSert for a specific role can have multiple guarantees for a single service differing in the strictness of the guaranteed deviation bounds from nominal behavior, it is possible to increase the chances that demands from other collaboration partners can be matched successfully at runtime. In a platooning scenario, multiple guarantees allow collaborations between more truck variants differing in their configurations, which means that even older trucks with a limited amount of deployed sensors could collaborate together. However, less rigorous safety guarantees consequently lead to more conservative worst-case behavior and thus the performance is worse, i.e. the inter-truck distance must be increased. In this way, ConSerts are an appropriate means for enabling a dynamic adaptation in changing runtime contexts.

4 Engineering safe nominal behavior

This chapter presents the series of activities that need to be carried out for the construction of a safe nominal behavior specification. It is structured according to *Figure 4.1*, which represents the sequence in which the activities have to be performed in reality. Each of the sections will be subdivided into two parts, where the first part describes the activities and concepts for OAS collaborations in general, whereas in the second part these concepts are applied explicitly to the truck platooning collaboration.

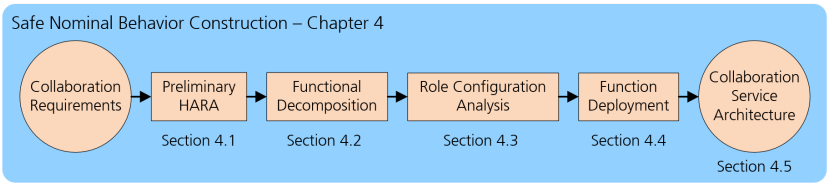


Figure 4.1: Safe nominal behavior construction – Chapter structure

4.1 Preliminary hazard and risk analysis

The first activity that has to be carried out to engineer a safe nominal behavior specification is the exploration of possible accidents for the platooning collaboration as well as their causes. As it has been outlined in *Section 3.2.1*, the goal of the preliminary hazard and risk analysis is to find hazards and accidents that have their cause in the intended functionality rather than in failures. These hazards and accidents are needed to construct a safe functional behavior during the absence of hardware or software failures.

4.1.1 Problems of conventional HARA for OAS

Initially, we tried to apply known hazard and risk analysis (HARA) techniques for truck platooning such as the one required by the ISO 26262 standard [27] where accidents and their potential causes are systematically collected in the shape of hazardous events (combination of a hazard and an operational situation). A fundamental assumption of this approach is that the considered hazards are unintended system behaviors, so-called malfunctions, which are deviations from the specified intended functionality of a system. Furthermore, it is assumed that a system is safe, if the risk of the relevant hazards is confined to an acceptable level by the ap-

propriate use of safety measures. Both assumptions sufficiently cover the prevention of accidents that are caused by behavioral deviations due to the occurrence of systematic software or random hardware failures within a system.

However, when trying to systematically determine malfunctions and their hazards for a collaboration of OAS, there appear two problems:

1. It is hard to clearly determine, if a behavior of an OAS is malfunctioning and thus safety-critical for the collaboration without considering the combination of behaviors of *all* other participating OAS, too.
2. The exploration of potential hazardous events does not only require the examination of behaviors and states of all collaborating OAS and the general environmental situation, but also their effects on the overall collaboration state. A systematic determination of a complete set of hazardous events is already a non-trivial task for closed systems and thus the extension of the scope to OAS collaborations leads to a combinatorial problem, when combining vehicle behaviors, vehicle states with possible environmental situations.

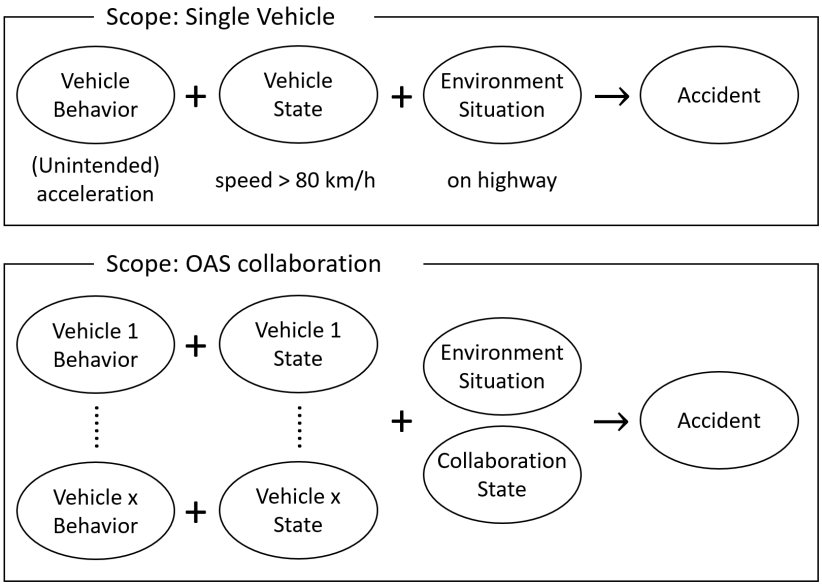


Figure 4.2: HARA scope single systems vs. OAS collaborations

Both of these problems are visualized in *Figure 4.2*: At the top, the situation is shown for a closed system like a conventional vehicle. A typical hazardous event in this case could be "an unintended acceleration maneuver on highway at speeds greater than 80 km/h", which may lead to an accident with other road users. Obviously, it is clear within the scope of a single vehicle that a sudden acceleration on the highway at high speeds

is dangerous and needs to be avoided. However, if a vehicle is participating as follower in a platoon (bottom of *Figure 4.2*), it is necessary to include the potential behaviors and the state of the leader vehicle as well. If for instance the leader truck is currently accelerating, an acceleration of the follower is not safety-critical but even required to maintain a constant separation. Thus, it is of high importance to see which effects a certain behavior has on the overall collaboration state to judge if it is safety-critical.

Due to the highlighted difficulties in evaluating the behaviors and states of all single OAS, the approach taken in this thesis is to focus on the overall collaboration only and ensure by design that dangerous collaboration states are excluded. In this way, we first have to find the accidents that should be avoided and subsequently determine those collaboration states leading to the accidents. According to *Section 2.3.2*, the only relevant accident that has to be considered for the running example is a frontal crash between follower and leader truck, since cut-in situations and lateral movements of any truck have been ruled out as a prerequisite in this thesis. Such a frontal crash can be caused by two scenarios:

1. The leader truck driver performs an emergency braking maneuver and the follower truck's full braking action is inadequate to prevent a frontal crash.
2. The follower truck accelerates to a higher speed than the leader truck and constantly keeps that higher speed until a crash occurs.

The essential difference between the scenarios is that the first scenario can be expected within a platoon as part of the intended functionality, while the second scenario cannot. It can always happen during platoon driving that the leader driver has to perform an emergency braking maneuver to which the autonomously driven follower truck has to react. The second scenario on the other hand contains an exceptional situation, because the goal of platoon driving is to maintain a certain distance, which is only possible *when both trucks have the same speed*. Thus, the scenario contains a follower truck behavior that deviates from its intended behavior and therefore does not need to be considered for the specification of the collaboration's nominal behavior. For this reason, we will only consider the first scenario in the following.

4.1.2 Platooning safe condition derivation

A suitable candidate for characterizing a platoon's collaboration state is its state of motion, more specifically the inter-truck distance d and the relative speed $v_{rel} = v_{leader} - v_{follower}$ between leader and follower truck. This means that the collaboration state might change as soon as distance or relative speed are influenced by acceleration or braking maneuvers of leader or follower. As stated above, the desired steady-state value for the relative speed in a platoon should be zero, so the only variable left for

modifying the collaboration state is the inter-truck distance. However, the distance can physically only be influenced by increasing or decreasing the relative speed, but this should only happen *temporarily* and thus does not clash with the general requirement that both platoon trucks should have the same speed during steady-state platooning.

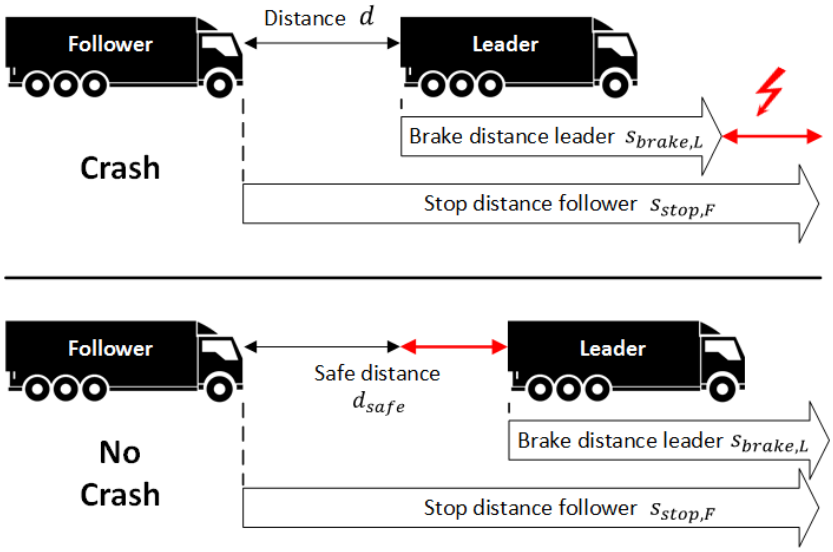


Figure 4.3: Safe distance visualization

Having chosen the truck separation as the physical variable representing the collaboration state, it is necessary to analyze the distance value range with respect to safety, i.e. which distances may lead to a crash and which are safe. Considering the emergency braking scenario introduced above, *Figure 4.3* shows two situations of a platoon at the beginning of an emergency braking maneuver initiated by the leader truck. The white arrows indicate the traveled distances $s_{brake,L}$ and $s_{stop,F}$ of both trucks at the moment where the trucks finally come to a stop, while the inter-truck distance d is the current distance immediately *before* the emergency braking maneuver starts. Note that the stop distance is defined as the sum of reaction distance and braking distance. From the point of view of the follower truck that needs to react to the braking leader, we need to explicitly consider its reaction distance, because during the reaction time, no follower braking happens, i.e. the inter-truck distance d is decreased. In contrast, the reaction time and distance of the leader truck is irrelevant since it is assumed that the follower's reaction time starts only at the moment, where a leader deceleration is perceivable for the follower.

Having introduced the required physical variables, we want to derive a physical *safe condition* that guarantees the avoidance of a crash, if evaluated to true. The upper part of *Figure 4.3* therefore delineates a situation,

where a crash will occur, because the stand-still point of the follower's front end is in front of the stand-still point of the leader's tail end, i.e. the follower truck is located inside the geometry of the leader truck (=crash). This crash can be avoided if the red indicated distance is added to d before the braking maneuver is started, which results in the follower stopping short of the leader (bottom of *Figure 4.3*). The distance needed to prevent a frontal crash will be called safe distance d_{safe} and is defined as:

$$d_{safe} = s_{stop,F} - s_{brake,L}$$ Safe distance definition

The safe distance value splits the inter-truck distance value range in two sets, a safe and an unsafe one. Obviously, all distances that are greater than the safe distance will also be safe and we can thus formulate the safe condition for truck platooning as:

$$d \geq d_{safe}$$ Platooning safe condition

Figure 4.4 summarizes the states of the platooning collaboration in a state machine. The *Safe Platooning* state represents all inter-truck distances d, where the safe condition is met, while the *Unsafe Platooning* state reflects its violation. Because of the fact that both $s_{brake,L}$ and $s_{stop,F}$ are highly

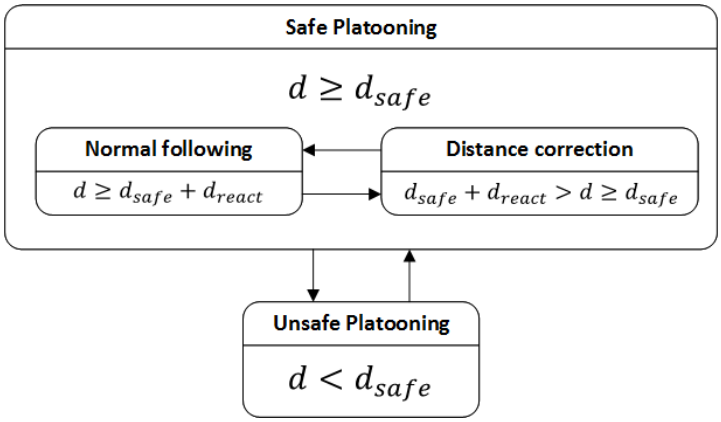


Figure 4.4: Platooning collaboration states

influenced by dynamically changing environmental and truck parameters, it is possible that d_{safe} changes as soon as one of the influential parameters changes. This would mean that if the platoon exactly maintains d_{safe}, there is no possibility to react to parameter changes without ending up in the unsafe state. Thus, the solution is to subdivide the *Safe Platooning* state in two sub states *Normal following* and *Distance correction*, where the former describes steady-state platooning while the latter reflects the temporary state where a *higher* distance set-point has to be realized. In order to perform distance adjustments in a safe way, it is necessary to maintain a higher distance than d_{safe}. The derivation of the exact value

of this additional distance d_{react} is out of this thesis' scope, because it is mainly dependent on how fast a specific controller design can realize a new distance set point, which is a problem located in the control theory area. For the sake of simplicity, d_{react} will be assumed to be a constant value representing a worst-case estimation.

The eventual goal of all further activities in *Chapter 4* will be to make sure by construction that the safe condition will always be met, provided that no systematic software or random hardware failures occur.

4.2 Functional decomposition

This section describes the functional decomposition of the platooning system based on the safe condition, i.e. to find the physical quantities, which need to be measured to determine the safe distance d_{safe} and how a given distance set-point d can be realized for a platoon.

4.2.1 Decomposition strategy

Figure 4.5 shows the initial functional model of the platooning system. *Safe Distance Computation* is the function on which the subsequent decomposition activities will be based on. From a functional perspective, its responsibility is to compute a safe distance set-point for the platoon by requiring follower stop distance and leader brake distance to be provided. Note that this function directly reflects the *Platooning safe condition* defined in *Section 4.1*.

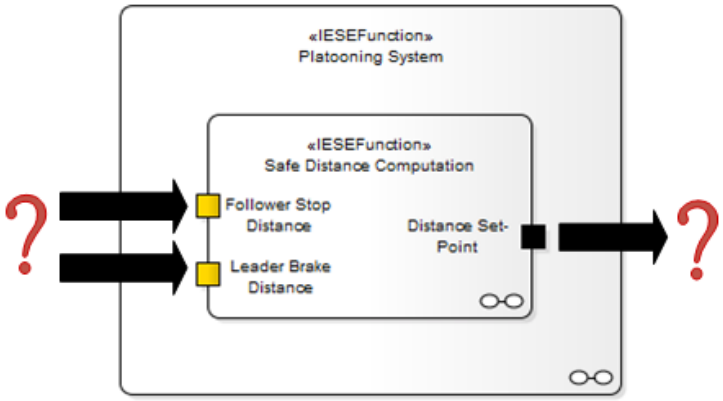


Figure 4.5: Initial functional model of the platooning system

Traditionally, the functional decomposition of a system is carried out until the behaviors of resulting sub functions can be expressed mathematically so that they can be directly implemented in code. However, the goal of

the decomposition in this section is different: We are only interested in the information *which* physical quantities the system needs to measure and actuate rather than *how* their exact formal relation to each other looks like. With two actual systems (follower and leader trucks) building the platoon, it is necessary to decide which functions shall be deployed to which truck. The fundamental idea of this thesis to make reasonable deployment decisions is to look at source and sink locations for system input and output variables. Thus, the platooning system has to be refined to the point where those basic physical properties needed to enable the collaboration in first place are known. Concretely, we need to determine on the one hand those physical input variables that allow a computation of follower stop distance and leader brake distance and on the other hand the output variables that enable a realization of a distance set-point computed by *Safe Distance Computation*.

The strategy that shall be used for the decomposition of the platooning system is referred to as *superposition* or *horizontal refinement* [30]. Its main difference to conventional decomposition strategies is that the decomposition does not start at the system output variables and proceeds against the signal flow. Rather, the horizontal refinement starts at some intermediate function (i.e. *Safe Distance Computation*) and refines in two directions, first *against* the signal flow until concrete input variables are determined and second *along* the signal flow until the system output variables are known.

4.2.2 Platooning system decomposition

This section shows how the platooning system is decomposed by making use of the horizontal refinement strategy starting at the inputs and outputs of the *Safe Distance Computation* function depicted in *Figure 4.5*.

Backward refinement towards system inputs

The follower stop distance $s_{stop,F}$ and the leader brake distance $s_{brake,L}$ are the physical quantities that have to be refined against the signal flow of the platooning system. The reason for a refinement is that both quantities are influenced by various other physical quantities and can therefore not be measured directly. In general, the brake distance is a sub quantity of the stop distance and thus their refinement includes the same considerations. In order to understand their relation, *Figure 4.6* shows the graphs of applied brake force, speed and traveled distance in the different temporal phases of a typical braking process.

The braking process of a truck can be subdivided in three parts:

Reaction phase $\mathbf{t_R}$ The reaction phase comprises the time span from the recognition of a danger until the actuation start of the brake system. For human drivers, t_R is highly dependent on the driver's skill and vigi-

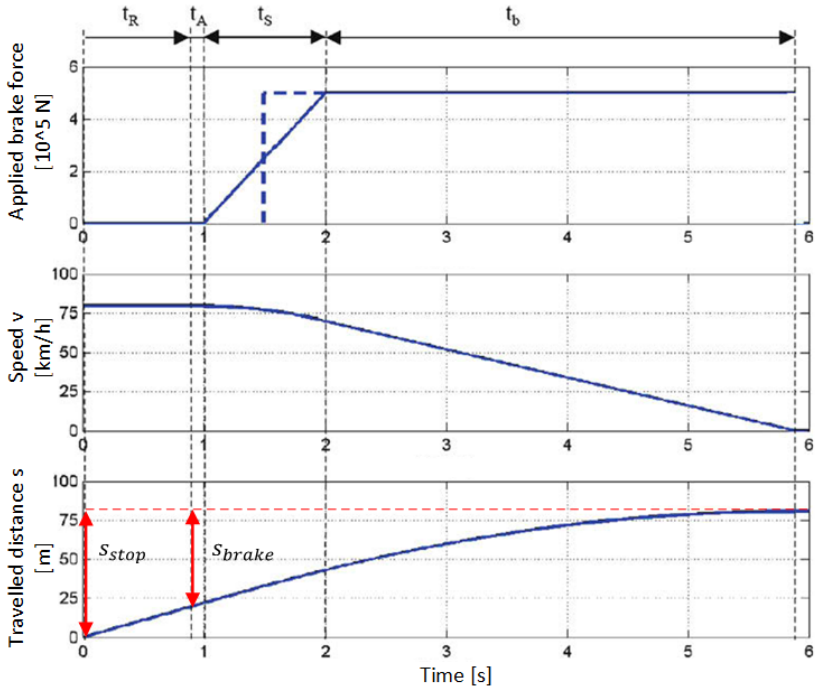


Figure 4.6: Temporal braking process (adapted based on [31, p. 113])

lance, while for autonomous systems, it is mainly determined by signal propagation and processing times until the brake is actuated. In both cases, no deceleration happens and the vehicle continues with its current speed v_0.

Dwell phase $\mathbf{t_A + t_S}$ The dwell phase comprises on the one hand the brake response time t_A, during which still no deceleration happens, because clearances within the brake system mechanics need to be overcome before any braking force can be exerted. On the other hand, the dwell phase contains the brake's dwell time t_S where the transferred brake force is increased until the maximum possible force. For the sake of simplicity, the brake force increase is assumed to be linear, although in general this might not be the case.

Brake phase $\mathbf{t_B}$ During the brake phase, the maximum available brake force is constantly applied until the vehicle comes to a stand-still ($v = 0$).

Having introduced the consecutive phases during the deceleration process, we can define the stop distance as the sum of traveled distances of all three phases, while the brake distance explicitly excludes the distance traveled during the reaction phase.

$$s_{stop} = s(t_R + t_A + t_S + t_B) \qquad \text{Stop distance definition}$$

$$s_{brake} = s_{stop} - s(t_R) \qquad \text{Brake distance definition}$$

With respect to platoon driving it makes sense to exclude the leader's reaction distance from the considerations, because the reaction time of the follower truck only starts as soon as leader braking is for the first time observable for the follower. Note that this can be the case already at the start of t_A, when the leader truck is able to communicate the brake pedal activation of its driver.

Based on the outlined brake process, [31] provides a closed form solution for the computation of the stop distance, if the constant maximum deceleration D_{max}, initial speed v_0 and characteristic values for t_R,t_A and t_S are given. The used braking model assumes that the applied brake force is proportional to the deceleration of the vehicle according to Newton's second law of motion $F = m \cdot a$. However, this assumption is too restrictive for the running example, because it neglects requirements that are demanded for realistic truck platooning (see *Section 3.1*), e.g. the consideration of the non-linear air drag or parameters for road conditions and truck characteristics, which turn the resulting vehicle deceleration into a non-linear quantity.

As mentioned before, the mathematical derivation of an explicit stop distance solution is not necessary at this point, because we only need to determine the system input variables[1]. Therefore, *Figure 4.7* gives a qualitative overview showing which basic physical quantities have an impact on the stop distance. This knowledge is sufficient for creating black-box functions where only the inputs and outputs need to be known.

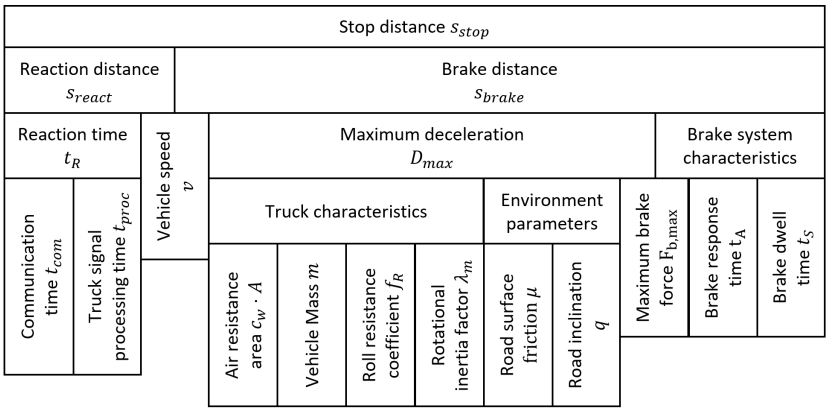


Figure 4.7: Physical quantities influencing the stop distance of a vehicle

All vertically annotated quantities are either dynamic variables that can be measured or determined by state-of-the-art or production sensors (t_{com}, v, μ, q) or they are parameters that are assumed to be static, at least for a certain driving distance ($t_{proc}, c_w A, m, f_R, \lambda_m, F_{b,max}, t_A, t_S$).

[1]The interested reader will find the mathematical derivation in *Section 5.2.2*, where it is finally required to build a simulation model.

Thus, the system inputs for the platooning system are clearly determined and the refinement in this direction is complete. The only missing task is the instantiation of the determined basic physical quantities for the leader brake distance $s_{brake,L}$ and the follower stop distance $s_{stop,F}$. The resulting functional model is shown on the left side of *Figure 4.8*.

Forward refinement towards system outputs

The refinement along the signal flow direction towards the system outputs (=actuated quantities) starts at the distance set-point output of the *Safe Distance Computation* function (see *Figure 4.5*). As mentioned earlier, the inter-truck distance can be influenced by a temporary change of the relative speed of the trucks, which can be realized by either an acceleration or deceleration of leader or follower truck. Within the platooning scenario, the follower truck is the one that autonomously reacts to driving maneuvers of the leader's human driver. Thus, the physical quantity that must be actuated by the platooning system is the follower truck's de-/acceleration. Being aware of the fact that deceleration and acceleration set points are typically realized by two different vehicle sub systems (brake and engine, respectively), we will nevertheless only use one system output *acceleration*, because first, this distinction is not relevant for the further considerations of this thesis and second, decelerations are from a physical perspective negative accelerations only.

Since the realization of a distance set-point is a typical control problem that has been extensively investigated in the area of adaptive cruise control (ACC) and similar advanced driver assistance systems (ADAS), a well-documented controller design from [32, p. 872 ff.] shall be reused in this thesis. In order to provide an acceleration set-point that is able to control on the one hand a specific distance set-point and on the other hand a relative speed of zero among the trucks, the current speeds of both trucks as well as the current distance between the trucks are needed as inputs for the controller. Note that the set-point for the relative speed stems from the fact that normal platoon driving demands both trucks to have the same speed. Since the speeds of the trucks have already been added as system inputs as part of the backward refinement, the only additionally needed system input is the current inter-truck distance d.

At this point, the horizontal refinement of the platooning system is complete. The final functional model is shown in *Figure 4.8*. On its left side, two functions *Follower Stop Distance Determination* and *Leader Brake Distance Determination* are given as a result of the backward refinement starting at the *Safe Distance Computation*, while the forward refinement yielded the *Distance Realization* function. Its output, namely the acceleration set point for the follower truck is the system output for the *Platooning System*, too. Note that the port labels end with "(F)", when a certain signal is related to the follower truck, while "(L)" indicates an association to the leader truck.

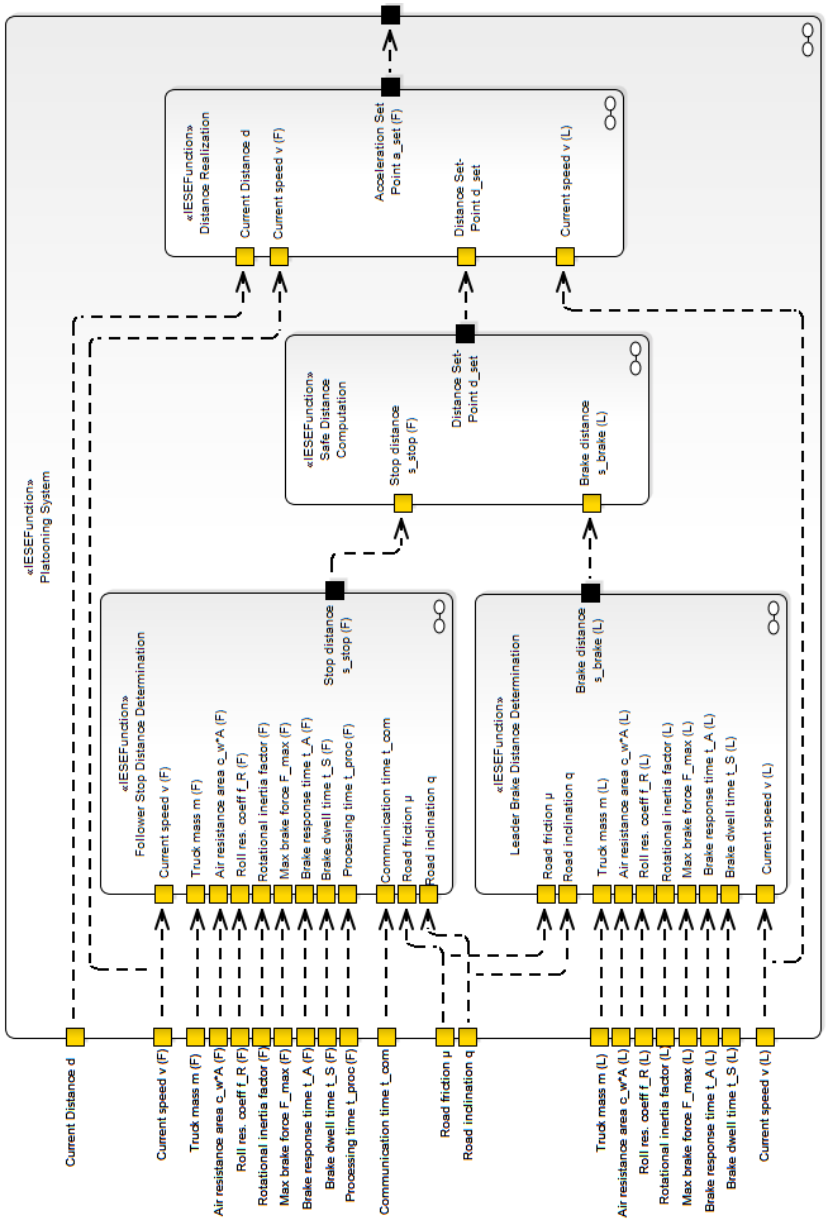


Figure 4.8: Final functional model of the platooning system

4.3 Role and configuration analysis

This section deals with an exploration of typical truck variants with respect to different types of variabilities, which have an influence on the function deployment and thus on the resulting service architecture as well. In order to cover truck variants in the context of a collaboration in a formal-

ized way, the concept of roles and configurations will be first explained in *Section 4.3.1*. Afterwards, *Sections 4.3.2* and *4.3.3* analyze the variants explicitly for truck platooning.

4.3.1 Role and configuration concept

In order to address variability on the collaboration level without considering concrete truck designs, an abstraction of the truck as a whole is necessary covering only that information which is relevant for the collaboration. A given truck can in principle support both the leader and follower roles during platooning, i.e. it can switch between the roles depending on its own and the collaboration partner's capabilities. Since the required functionality for the platooning collaboration is very different with respect to different roles, it is meaningful to use the collaboration role as a principal means for abstraction instead of directly dealing with the whole complexity of a given truck. On the basis of role definitions, it is possible to specify variants that may occur for a specific role by using so-called configurations. A configuration is mainly defined by its interface in shape of provided and required services (see *Section 2.2.2*) and the service interface will eventually result from a specific function deployment.

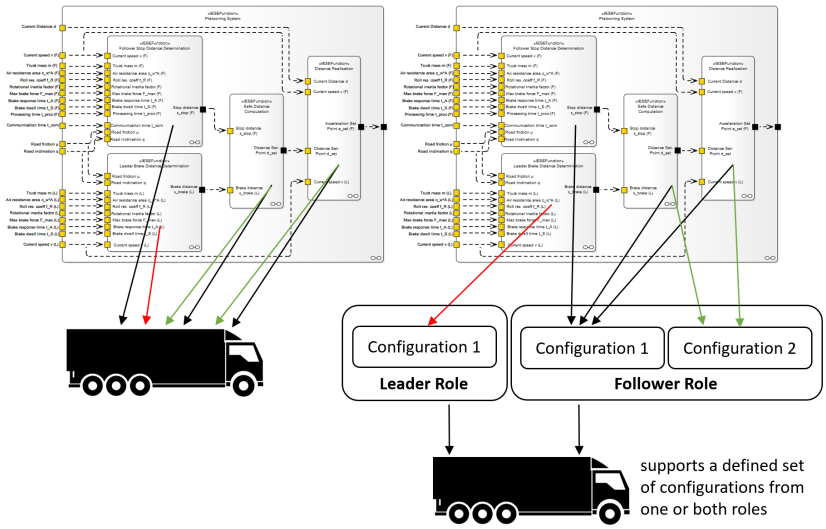


Figure 4.9: Variability support through roles and configurations

Figure 4.9 exemplifies the problem from the point of view of a truck manufacturer who wants to make an existing truck model platoon-ready. The arrows represent a deployment of a certain function and different arrow colors mean different possible variants. Given only the functional model as defined in the previous section, the manufacturer has to answer the following questions:

1. Should the truck be able to act only as follower or leader or even both roles?
2. Which functions need to be realized to support a certain role?
3. Is the current hardware equipment of the truck sufficient for platooning and if this is not the case, is there the possibility of choosing between variants for which safe platooning is guaranteed?

Without a clear structure that covers variability information on the collaboration level, it will be hardly possible to answer those questions (*Fig. 4.9*, left side). However, if several configurations for both roles have been pre-engineered based on common knowledge on typical variabilities and if the pre-engineered configurations have been explicitly certified with respect to safety, the manufacturer is very flexible in choosing a set of supported role configurations that fits best the existing truck design as well as the business goals of the company (*Fig. 4.9*, right side).

4.3.2 Platooning configuration analysis

Having introduced a concept for covering variability by means of roles and configurations, this section explains, which variability types exist and how to come up with a set of meaningful pre-engineered configurations.

A major result of the functional decomposition has been the knowledge about system inputs and outputs that are needed to enable a safe platooning collaboration. Thus, when deriving potential configurations for both follower and leader roles, an important consideration is on the one hand whether the required system inputs can be measured by existing sensors and on the other hand whether there are actuators that allow the realization of the system outputs. In this way, the first identified variability is the general **availability of software/hardware** within a truck for measurement and actuation. For the truck platooning scenario, the availability of an actuation mechanism, i.e. the realization of de-/accelerations, is clear since the presence of both engine and brake systems is a hard requirement for every meaningful truck operation. It is also clear that the actuation will be carried out on the follower truck, since the follower is the autonomous system during platoon driving. Thus, we will mainly consider in the following the availability of sensors that can explicitly support truck platooning in any way. Note that also software availability has been mentioned, because first, modern sensors typically contain software for post-processing measured raw data and second, software can be used for the determination of physical quantities that are hard or impossible to be measured directly by applying sensor fusion on the software level.

After having knowledge on the potential availability of physical quantities, the second type of variability is the set of **value characteristics** that a sensor is able to deliver:

Value accuracy The accuracy indicates, how accurate a measurement reflects the real physical value. Due to the used measurement principle, environmental noise, analog-digital conversion and other factors, there is always a certain error with respect to the real value. The maximum error and thus the lower and higher bounds of a value are determined through extensive testing and are therefore known for a specific sensor.

Value range The value range that a sensor can cover for a physical quantity. Depending on the usage scenario, only subsets of value ranges are required.

Value resolution The value resolution or quantization determines the distance between to subsequent possible values. Measurements falling in between two of these values have to be mapped to either of them, which is typically done by choosing the nearest possible value. Note that a certain value resolution also affects the value accuracy.

Based on the discussed variabilities, it is necessary to have a look at sensors that typical modern trucks are equipped with, but also measurement principles and sensor concepts that fall within the state of the art. The idea for pre-engineering potential configurations is to use the acquired knowledge and derive sensor deployments that are likely to occur and explicitly support platooning. In this way, it is possible to offer configurations ranging from trucks having a comprehensive sensor set to trucks with minimum sensing capabilities. The inclusion of state-of-the-art sensors even enables the support for sensor deployments that are not on the market yet and thus also give a guidance where further research is still required.

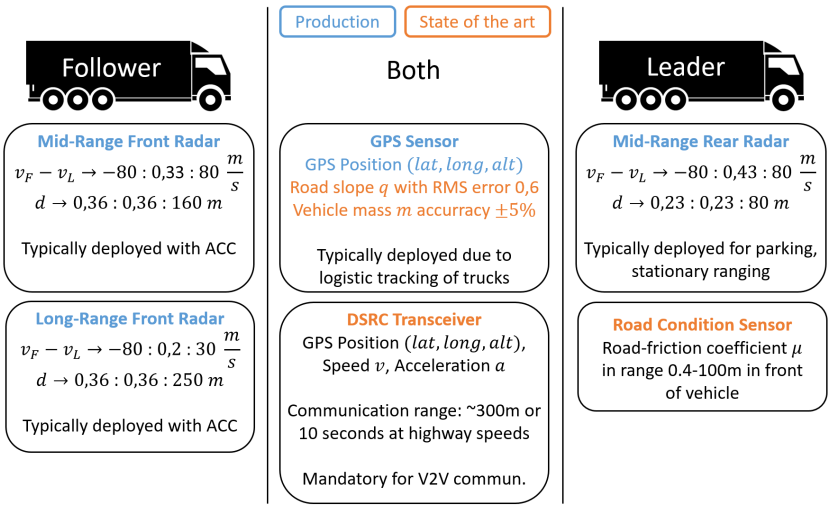


Figure 4.10: Platooning sensor deployment analysis

Figure 4.10 shows the sensors that have been found as result of a literature and market research. Note that the figure only shows some typical

representatives of the actually available sensors to illustrate the concept and therefore does not claim completeness of any kind. The depicted sensors mainly focus on those physical system inputs of the platooning system (see *Figure 4.8*) characterizing the collaboration and its environment (inter-truck distance d, relative speed $v_F - v_L$, road slope q, road friction μ). In contrast, the other required system inputs are either fixed parameters or physical quantities that relate to a single truck only. Thus, it can be assumed that these values are available in the respective truck. By having a look at the different kinds of presented radar sensors [32, p. 300f.], it can be observed that these are already used in production, either in adaptive cruise control systems (ACC) or for low speed ranging. This is no surprise since ACC is similar to truck platooning, but is constrained to single following vehicles and thus lacks of V2V communication. Concerning front vs. rear radars it is interesting to note that the rear radar has a higher value resolution for the distance measurements, which is beneficial for platooning. From the platooning perspective, the radar sensors should mainly measure the inter-truck distance and therefore it does not make sense to consider a front radar on the leader truck. GPS sensors typically deliver only the vehicle's position and the vehicle's speed, but [33] correlated the GPS data with vehicle internal data to estimate both vehicle mass and the road slope with good accuracy. Note that environment characteristics like the road slope are fixed for a certain position and thus this information could be retrieved by precise high-resolution maps as well. Without a communication transceiver on both trucks, V2V communication and thus collaborations are impossible. Since the *Dedicated Short Range Communication* (DSRC) standard [9, 11] explicitly addresses V2X communication, it can be considered as a reference protocol stack for automotive collaborations in general. Note that the DSRC transceiver delivers also relevant values like GPS position, speed or acceleration values in addition to mere communication infrastructure. This stems from the fact that these values are part of the standardized basic message that has to be provided by a vehicle using DSRC (see [10, 12] for more details on basic message sets). Another physical quantity that is in particular difficult to determine accurately is the road friction that is highly relevant for safe platooning, because it has a major impact on the stop distance. Continental developed a sensor fusion concept that allows the estimation of weather conditions and their impact on road friction until 100m ahead of a vehicle [34].

4.3.3 Platooning configuration selection

Based on the gathered knowledge on both sensors being used in production and sensor concepts falling known in the state-of-the-art, different configurations can be built for the leader and follower roles. When combining different sensor/actuator deployments for leader and follower it is of essential importance that *all* system inputs and outputs of the platooning system are covered. In realistic settings, a high number of sensor deployment combinations can be built and this number is even increased

when additionally considering the sensor value characteristics. Therefore the goal is not to provide a configuration for each possible scenario, but to choose the most probable and meaningful scenarios with respect to platooning. Since this choice highly depends on expert knowledge and involves negotiation among the companies of a domain, this thesis picks two random configurations for each role being capable of illustrating the further described concepts, namely how different configurations can yield different function deployments and therefore different service interfaces.

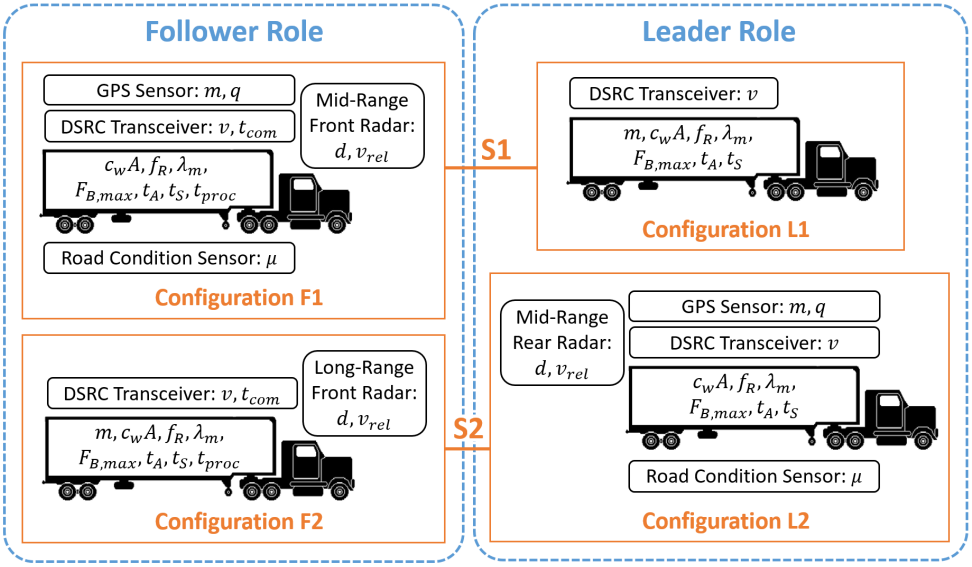


Figure 4.11: Platooning roles and their configurations

Figure 4.11 shows the chosen truck configurations for both follower and leader roles. A scenario exemplified for instance by S1 or S2 of *Figure 4.11* is a combination of exactly one follower and one leader configuration. As stated above, platooning can only be realized successfully for a certain scenario, if all system inputs and outputs are covered. This is for example not the case for the combination of configurations F2 and L1, since the road friction μ and the road inclination q are not available. In contrast, the scenario F1-L2 would enable platooning with very good performance, because a lot of redundancy is given. Since the consideration of all possible scenarios does not yield new insights, only scenarios S1 and S2 shall be considered in the following. From a practical point of view, it is on the one hand necessary to provide configurations for each role that cover a high range of potential truck variants. On the other hand, it is equally important to create configurations for both roles that can successfully realize platooning together.

In all configurations, the depicted platooning-specific sensors are annotated with those quantities that they can provide, while the annotation of

quantities inside the truck boundaries are assumed to be provided by the truck platform itself. Scenario 1 can be characterized by a follower truck that is equipped with a modern sensor set, while the leader truck only has minimal sensing capabilities. In scenario 2, the situation is inversed except that we have redundancy capabilities with respect to the measurement of the distance d and the relative speed v_{rel}. In this way, redundancies can be used for plausibility checks to deliver accuracy improvements leading to better functional performance during collaboration. Another aspect for the selection of configurations can be the cost of components. A mid-range front radar is likely to be cheaper than a long-range front radar, although both measure the same quantities, but with differing accuracies. Thus, it is necessary to quantify the impact of such a choice, because a more accurate sensor will lead to a smaller possible inter-truck distance during platooning, which in turn improves the saved fuel due to aerodynamic drag reduction.

4.4 Function deployment

Taking the functional model of *Section 4.2* as a starting point, the role and configuration analysis delivered variants for both follower and leader roles by linking the system inputs and outputs of the platooning system to the respective roles based on combined variants of available sensors. A prerequisite for defining the service interface completely is to look at different possible scenarios separately, i.e. specific combinations of leader and follower configurations and to deploy the platooning system's *inner* functions to the respective roles. This is in particular important for a truck manufacturer who wants to realize a certain role configuration, since the deployment of a function to a configuration forces the manufacturer to implement that functionality, which is in the first place a cost factor. Obviously, the decision for a deployment of an inner function to one or the other role should not be at random, but rather follow a dedicated strategy. Another desirable goal is that the deployment can be carried out systematically so that this process can be supported by tools in a (semi-)automatic way. In this way, *Section 4.4.1* explains a systematic strategy for function deployment that makes use of a set of heuristics. Afterwards, *Section 4.4.2* applies these heuristics to the two platooning scenarios created in *Section 4.3.3*.

4.4.1 Deployment strategy

The function deployment can be considered as an optimization problem whose solution should represent an optimal distribution of the inner functions of a system among the available collaboration roles. A challenge in this respect has been to find the criteria that characterize an optimal solution for the given problem. Logically this requires to start with an examination of the effects of certain deployments on the overall collaboration.

A distribution of related functions to different roles inevitably yields signal flows between roles, which will manifest themselves as provided and required services, since a role is implemented by exactly one OAS and cannot be spread over multiple OAS. Hence, the deployment has an impact on the number of services and therefore on the amount of data that has to be exchanged between the collaborating OAS, too. In general, it can be stated that the more equal the distribution of functions over the participating roles is chosen, the higher the number of resulting services will be. Put differently, if a single role realized the majority of functions, it would have maximum context knowledge and the dependencies to other roles would be minimal, which represents the ideal case. However, the essential nature of collaborating OAS is that their (different) capabilities are combined to a new behavior and this combination is only possible through services. Therefore, services cannot be avoided per se, but it is nonetheless desirable to have roles with maximum and minimum responsibilities. Apart from this collaboration-level consideration, the existence of a service can be evaluated from a technical perspective, too:

1. The wireless transport of information from source truck to target truck has a time delay compared to the (typically) closed wired networks like CAN within a single truck.
2. The wireless communication medium is not exclusively used by the collaborating trucks, but also by other road users. Thus, it is harder to guarantee successful and timely communication, because the communication medium is dynamically accessed by an unknown number of vehicles.
3. Compared to a provision of information from the own truck platform, the external dependency to another OAS through a required service cannot be as trustworthy with respect to the guaranteed information quality, because the service provider is not known at design time.

Based on these findings, a reasonable strategy for the deployment is to minimize the number of services and hence the existing dependencies between collaborating OAS, too. In addition to the consideration of a service as an abstract dependency to another OAS, it proved useful to also take into account whether a service transfers continuous signals or if the carried data is transferred sporadically only, because it changes rarely or not at all during collaboration. By virtue of their nature, continuous signals are much more influenced by timing delays than sporadic or static signals. In addition, they need to be transferred with a high frequency and therefore stress the communication medium. Thus, the deployment strategy should prefer services carrying sporadic data over services carrying continuous data.

The idea of this thesis for performing a systematic function deployment to roles is to start at the platooning system's inputs, whose source roles have already been fixed as part of the configurations and proceed along the signal flow through the platooning system.

For each visited function, the deployment role is chosen based on a set of heuristics. As soon as the function's role is fixed, the system is further traversed through the function's outputs. The heuristics compile the above findings about an optimal deployment into rules that can be decided by examining the count and characteristics of the function's inputs and outputs. The process is finished, when all inner functions have been successfully deployed to a role. The deployment heuristics for the evaluation of a given function are presented in *Table 4.1*.

H1	If *all* function inputs originate from the same role, deploy the function to that role.
H2	Count the number of continuous inputs for each role. If an input is available at multiple roles (redundancy), count it for the role delivering more accurate data. Deploy the function to the role where the majority of continuous inputs originate from.
H3	If the number of continuous inputs is equal for each role, count the sporadic inputs for each role. Deploy the function to the role where the majority of sporadic inputs originate from.
H4	Increase the amount of responsibility of a role, if more than 50% have been already deployed to that role.
H5	Functions whose responsibility is to directly control an actuator continuously should be co-located to the actuator and thus be deployed to the same role that provides the actuation.
H6	If a function could not be deployed clearly with H1-H5, either consider a further vertical refinement of that function into multiple smaller functions or examine the consequences of a certain deployment with respect to other quality attributes.

Table 4.1: Function deployment heuristics

4.4.2 Platooning function deployment

This section deals with the practical application of the function deployment concept that has been presented in the previous section. As input products, the functional model (*Figure 4.8*) as well as the role configuration model (*Figure 4.11*) of the platooning system shall be used. Although the four given configurations yield three potential collaboration scenarios, the heuristics are only exemplarily applied to the indicated scenarios S1 and S2. The procedure will be to start at the system inputs and to apply the heuristics given in *Table 4.1* systematically to each function along the signal flow.

Figure 4.12 shows the functional model annotated with the relevant information from both configurations of collaboration scenario 1. On the one hand, the annotations include a difference in color to show the origina-

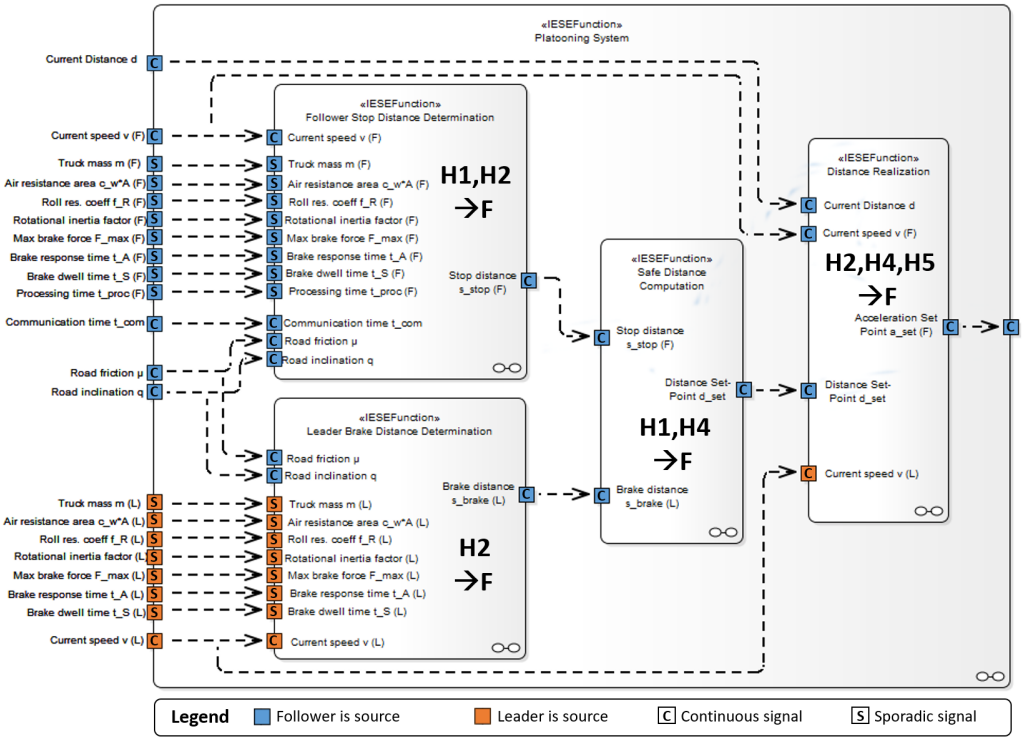


Figure 4.12: Function deployment – Platooning scenario S1

tion of a specific input or output. Blue inputs are available through truck platform or sensors of the follower truck, while orange inputs mean the same for the leader truck. The letters **C** and **S** on the other hand indicate whether a certain port is source or target of a **c**ontinuous or **s**poradic signal. Note that the figure already depicts the final result after the deployment; at process begin, only the system inputs and outputs of the *Platooning system* function are annotated.

The first examined function is *Follower Stop Distance Determination*, whose inputs assume their annotation from their respective system inputs through the signal flow. A precondition for most of the heuristics is to count the amount of continuous inputs for each role. The reason why the sporadic inputs are excluded from the consideration at first is that services carrying continuous signals should be avoided in the first place. Afterwards, the counted input amounts of both roles are compared and since the given function only has inputs originating at the follower role, the follower amount of three is obviously higher than the leader amount of zero and therefore the function is deployed to the follower role according to heuristic H2. In order to save the counting process for functions having only inputs originating from a certain role, heuristic H1 has been added to be able to directly deploy the function to that role. After the deployment decision has been taken, the outputs of the deployed function

automatically assume the source annotation, i.e. the *Stop Distance* output is now originating from the follower and propagates its affiliation again through the signal flow to the next function *Safe Distance Computation*.

Since not all inputs of *Safe Distance Computation* have an annotation and therefore the heuristics cannot be applied yet, we look at *Leader Brake Distance Determination* first. Heuristic H1 does not yield a decision here, so the inputs have to be counted according to H2. The majority of continuous inputs originates from the follower and thus the function is also deployed to the follower. An interesting observation is that although there are eight leader inputs and only two follower inputs, the function is still deployed to the follower role. The reason for this decision is that out of those eight leader inputs, seven of them are carrying sporadic signals that are either static over the whole collaboration, because they are parameters or only rarely change.

The *Safe Distance Computation* function can be clearly deployed to the follower role due to heuristic H1. In addition, heuristic H4 supports this choice, because as yet, two out of four inner functions (=50%) have been already deployed to the follower role. Because of the fact that we strive for roles with maximum context knowledge, H4 gives an indication with respect to this goal. Note that H4 should be understood more as a guidance criterion for the engineer than as a definite rule, because the choice of a role with maximum responsibility could be influenced as well by other (quality) aspects like safety.

Having three functions deployed to the follower role the last undeployed function is *Distance Realization*. Again, heuristics H2 and H4 give a clear indication for the deployment to the follower, but in addition, heuristic H5 is valid in this case, too. Since the examined function's output is directly the continuous control set-point for brake and engine which both are deployed to the follower it is in particular important that the function is deployed to the follower as well. The reason behind this statement is that the follower actuation is from the system perspective the only possible means to *actively* change the collaboration state and hence it is a functionality that should not be exposed to an additional failure risk due to external communication through a service.

At this point, all four inner functions have been successfully deployed and thus the process is finished. The fact that all functions and hence the whole responsibility have finally been deployed to the follower role is not a coincidence. By having a look at the examined collaboration scenario (S1 in *Figure 4.11*), it can be observed that the follower truck delivers most of the required quantities plus the actuation mechanism and it is therefore a reasonable idea to assign the main responsibility to the follower role as well. For our rather small platooning system, this initial "guess" based on looking at the configurations might lead to the right choice in the end, but if collaborations are considered that have larger sizes in terms of roles, func-

tions and inputs, it is worthwhile to leverage from the presented heuristics for the substantiation of the engineer's guesses.

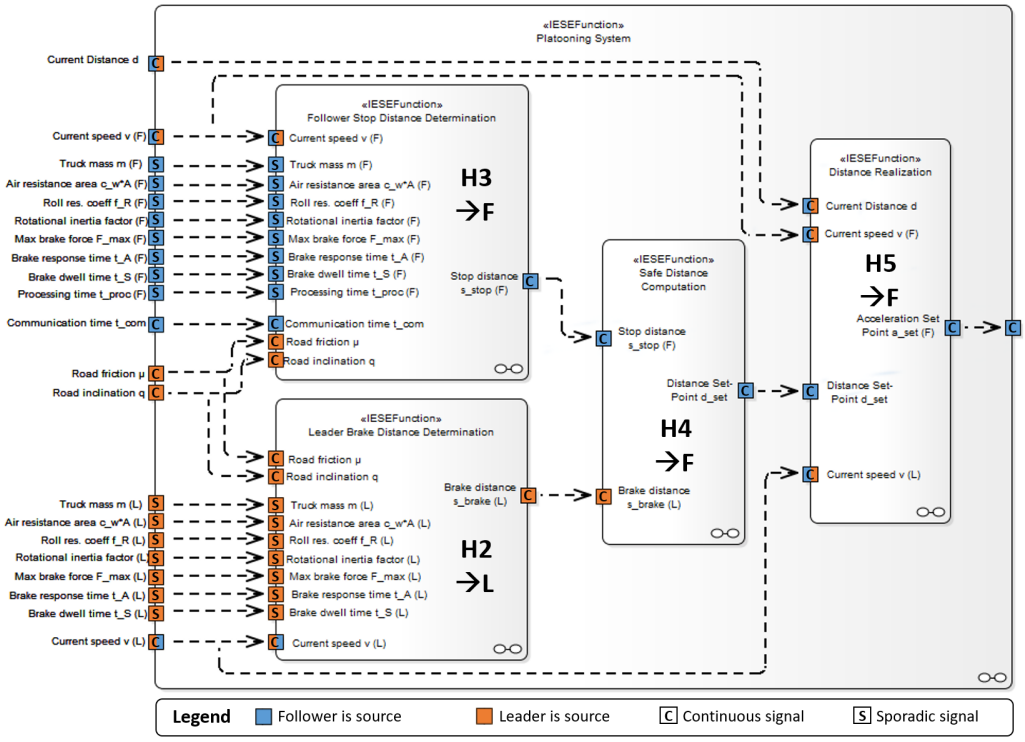


Figure 4.13: Function deployment – Platooning scenario S2

In order to see another example for the heuristic application, *Figure 4.13* shows the annotations of the functional model based on the configurations of collaboration scenario S2 (see *Figure 4.11*). One difference to the first scenario S1 is that some of the system inputs originate at both roles due to the redundancy of the sensor deployment. Thus, there is more flexibility during deployment, because although redundant inputs might have differing accuracies with respect to their signals, there is still the possibility to choose one over the other for the sake of avoiding newly introduced services during deployment.

For the *Follower Stop Distance Determination* function, we chose the follower origination for the *current speed*, because the truck-internal speed value is by far more accurate than a speed value that is computed at the leader truck by measuring the relative speed and subtracting it from the leader's internal speed. In this way, the counting of continuous inputs per role according to heuristic H2 yields equal amounts. In this case, the sporadic inputs are considered, because a deployment based on H2 would yield the same amount of services. Since only follower sporadic inputs are

existent, the function is deployed to the follower role according to heuristic H3.

In case of *Leader Brake Distance Determination*, heuristic H2 advocates that the function should be deployed to the leader role, independent of how the redundant input *current speed* is resolved here. In order to avoid a new service, the leader role is chosen as a source for this input.

Due to the facts that both inputs of *Safe Distance Computation* have been deployed to different roles, that there are no sporadic inputs and that one role does not have more responsibility than the other yet, heuristics H1-H4 are not applicable. However, what is known is that the *Distance Realization* function has – like in the first example – a direct output to the actuation part of the follower and should thus be deployed to the follower according to H5. This decision allows maximum flexibility for the resolution of the input redundancies of *Distance Realization*. The current distance and current speed follower's are chosen to be originating at the follower role, while the leader's current speed is chosen to originate at the leader role for accuracy reasons. Finally, *Safe Distance Computation* can be deployed according to heuristic H4, because there is a 2:1 deployment ratio for the follower role.

In summary, the presented function deployment examples yielded similar results with respect to the role taking the major responsibility (follower), but in the second scenario, one function was chosen to be more favorable to be deployed to the leader role. Based on the results of the deployment, the next section will explain the resulting service architecture for the platooning system.

4.5 Service architecture derivation

At this point, all necessary engineering activities have been performed for the construction of a safe nominal behavior specification for the platooning system. This section assembles the previous result products into a big picture – the service architecture. Therefore, *Section 4.5.1* explains first, how a generic service architecture looks like for a collaboration of OAS. Afterwards, the service architecture for the platooning system will be described in *Section 4.5.2*.

4.5.1 Generic collaboration service architecture

The generic service architecture for OAS collaborations is illustrated in *Figure 4.14*. It relates all concepts and artifacts with each other that have been mentioned in this chapter so far, namely the scenarios and roles of a collaboration, the configurations of a role, the provided and required services of a configuration and the realization of provided services through deployed functions.

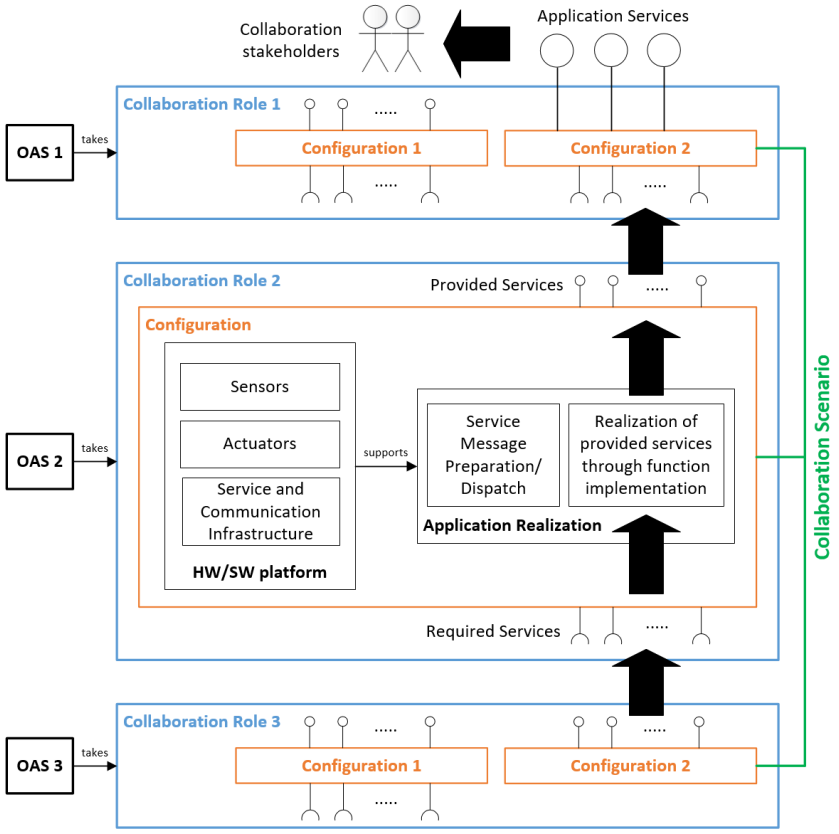


Figure 4.14: Generic OAS collaboration service architecture

For a planned collaboration, the overall goal demands that different OAS being developed in isolation can safely collaborate with each other at runtime. Since each OAS can only take exactly one collaboration role at a time, at least as many OAS as required roles are needed for a successful collaboration.

A role can have multiple different configurations that represent different pre-engineered variants and thus enable adaptivity due to the possibility to dynamically switch between configurations during runtime. The covered variabilities of the pre-engineered variants concern the hardware and software platform, i.e. configurations guarantee that the underlying HW/SW platforms provide different combinations of available physical quantities and their accuracies. Note that a configuration only sets up requirements directly targeting physical quantities instead of concrete sensor, actuator and software designs. Hence, there is still flexibility for the concrete realization of the required accuracies on the implementation level.

The idea for the upfront validation that a collaboration is safe during the absence of random hardware or systematic software failures is to look

at specific combinations of role configurations, the so-called collaboration scenarios. Based on the fixed configurations of such a scenario, the overall collaboration functionality is distributed among the roles. Thus, when two functions with a signal flow dependency in between are deployed to different roles, the functional dependency is subsequently a dependency between configurations of different roles. Such inter-role dependencies are called services, which decouple different roles from each other to enable their realization in isolation. Note that services are associated to configurations and not to roles, because a role can offer multiple configurations and a service interface results from a function deployment due to the consideration of a specific configuration. However, the terms *role* and *configuration* can be used as synonyms during the consideration of a specific scenario, because only one configuration can be active at a time for a role.

In this way, a service hierarchy is constructed, in which the required services of one role configuration are satisfied by configurations of other roles (the path of black arrows in *Figure 4.14*). On the one hand, the hierarchy ends at configurations that can provide their services without requiring further services from somewhere else. On the other hand, the top or root of the service hierarchy consists of provided services that have a special nature in that they are not providing services to other systems, but to the stakeholders of the collaboration, which are typically human-beings. These services are called application services and include more abstract guarantees, e.g. safe platoon driving for the running example. The main reason for the explicit definition of application services is the necessity to assign a responsibility for the collaboration goal achievement to a certain role. Since safety is always a goal of safety-critical collaborations, the responsibility assignment is in particular important for safety considerations as we will see during the definition of ConSerts in *Chapter 5*. Concerning the choice of a role guaranteeing the application service, it turned out to be reasonable to select that role having the highest context knowledge, which is typically the role being at the top of the service hierarchy.

Given that a collaboration service architecture has been created and documented on the domain-level including the artifacts described above, it shall now be described, how an OAS manufacturer can use this information to actually implement or extend a system that will be able to successfully collaborate with other OAS. The manufacturer has to decide first, which role(s) his OAS should be able to take. From the role description, which contains a list of the potential configurations, at least one has to be selected. A configuration description compiles the following relevant information:

1. The physical quantities that have to be available in the system with a specified accuracy.
2. The services that have to be provided with a given accuracy to other OAS.

3. The services that other OAS will provide to the given system (=required services).
4. The functions that have to be implemented in order to provide the configuration's services
5. Information on the potential functional performance, when the configuration is implemented. Different configurations typically yield different performances in different collaboration scenarios due to varying capabilities.

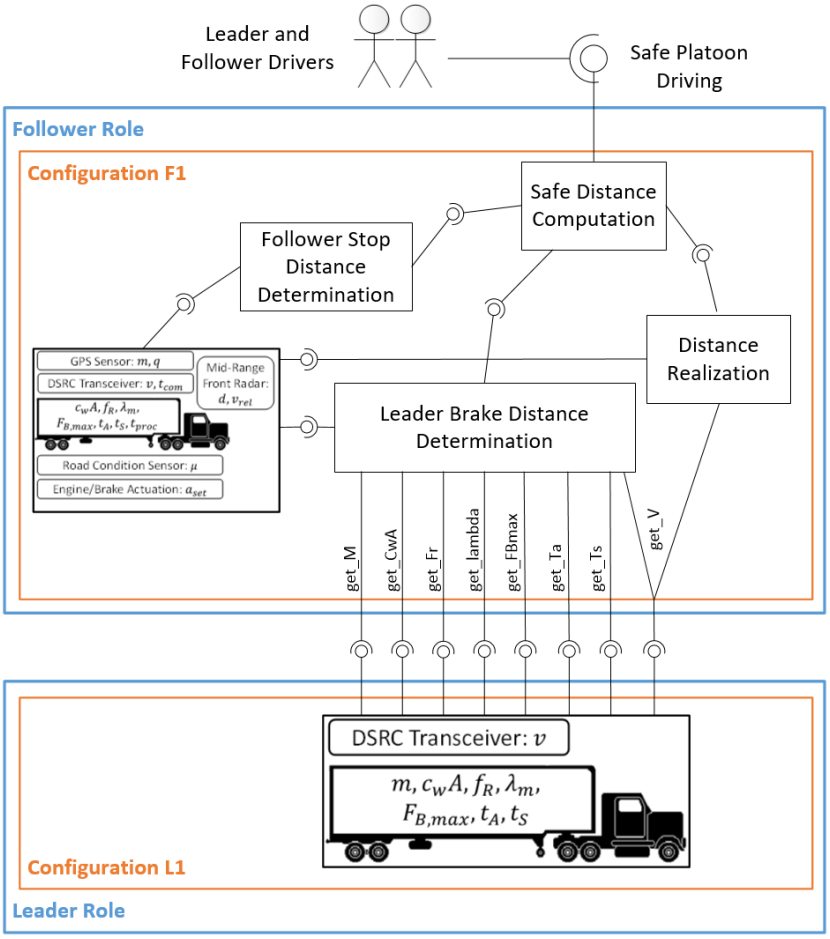
The decision for the selection of one or more configurations involves a trade-off between cost, performance goals and the flexibility to achieve a successful collaboration with as many other OAS as possible. In order to achieve this flexibility, the number of implemented configurations must be increased. In case an existing OAS should be extended to support a new collaboration type, the existent hardware/software platform also plays an important role for the configuration selection. An eventual performance increase has to be critically weighted against the cost of knew hardware.

Finally, the manufacturer needs to realize the provided services by implementing the functions associated to the selected configuration. This includes not only the application logic, but also the preparation and dispatch of service messages, if this is not taken care of by a standardized service communication framework.

4.5.2 Platooning service architecture

This section illustrates the service architecture for both collaboration scenarios of the platooning system. The required input products are the functional model (*Figure 4.8*), the collaboration scenarios documented in the role configuration model (*Figure 4.11*) and the function deployment to roles (*Figures 4.12* and *4.13*). *Figures 4.15* and *4.16* show the service architectures for the collaboration scenarios S1 and S2.

Scenario S1 consists of both follower and leader roles with the active configurations F1 and L1. The inner functions of the platooning system have been deployed to the respective roles according to *Section 4.4.2*. An important aspect that has not been explained in detail in the deployment section is how service interfaces between roles are identified in principle based on a specific deployment. In scenario S1, all inner functions have been deployed to the follower role and it can be observed that not all required inputs of these functions can be provided by the follower truck platform (indicated with the truck on the left side). Since no other roles are existent for platooning, these inputs have to be provided from the leader role through services. The functional service types of these services are defined more detailed in the table at the bottom of *Figure 4.15*. Since the service type name can not hold enough information, the table specifies in detail the physical quantity for each functional service type including



Service Type	Description	Unit	Range	Quantization	Minimum Frequency
get_V	The current longitudinal speed v of the vehicle (Continuous)	$\frac{m}{s}$	$0 \leq v \leq 40\frac{m}{s}$ $40\frac{m}{s} = 144\frac{km}{h}$	$0.01\frac{m}{s}$	10 Hz
get_CwA	The air resistance area $c_w \cdot A$ that is relevant for calculating the air resistance force (Sporadic)	-	$0 \leq c_w \cdot A \leq 22.5$ $0 \leq c_w \leq 1.5$ $0 \leq A \leq 15m^2$	0.01	At platoon building
get_M	The overall vehicle mass m, which includes truck and truck load (Sporadic)	kg	$0 \leq m \leq 50t$	$1kg$	At platoon building
get_Fr	The roll resistance coefficient f_R Describes deformation losses between tires and road (Sporadic)	-	$0 \leq f_R \leq 0.05$	0.001	At platoon building
get_FBmax	The maximum brake force $F_{B,max}$ that the brake system can deliver with optimal conditions (Sporadic)	kN	$0 \leq F_{B,max} \leq 500kN$	$0.01kN$	At platoon building
get_Ts	The brake system's dwell time t_s the brake needs to deliver the maximum brake force in optimal conditions (Sporadic)	ms	$0 \leq t_s \leq 500ms$	$1ms$	At platoon building
get_Ta	The brake system's response time t_a the brake system needs to overcome mechanical clearances (Sporadic)	ms	$0 \leq t_a \leq 250ms$	$1ms$	At platoon building
get_lambda	The rotational inertia factor λ_m that considers rotational inertia parts contributing to the acceleration resistance (Sporadic)	-	$1 \leq t_s \leq 1.6$	0.001	At platoon building

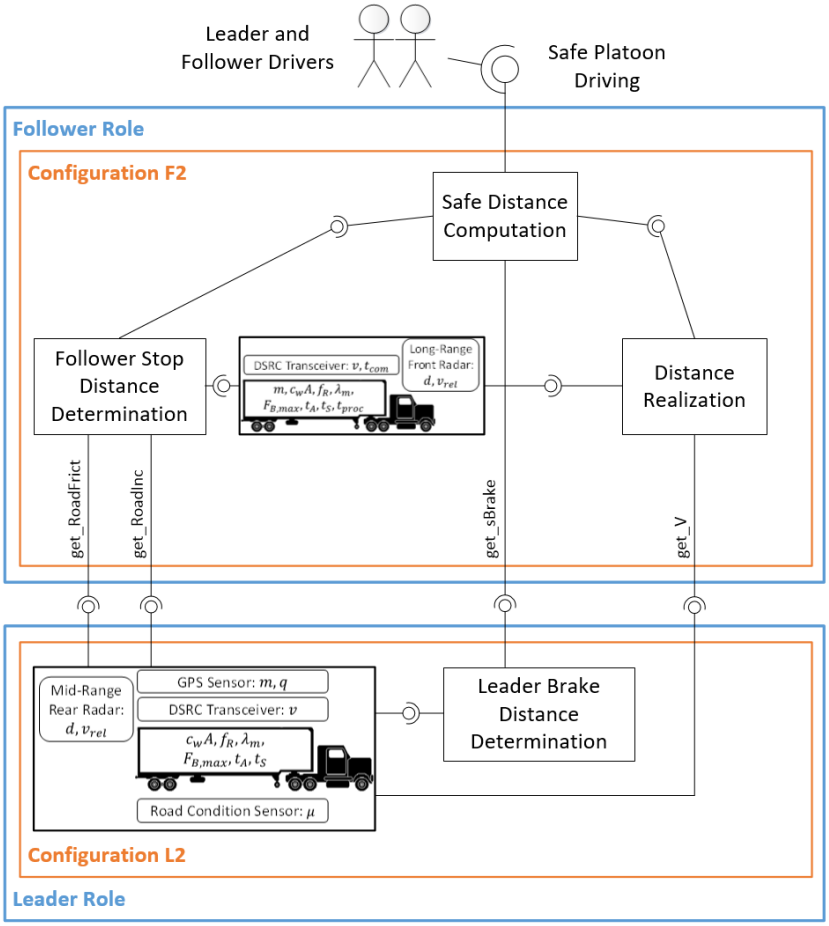
Figure 4.15: Service architecture - Platooning scenario S1

description, unit, expected value range, quantization and a minimum frequency indicating how frequently the service will be used. Out of the eight required services, only the service *get_V* carries a continuous signal while all other services are only used once at platoon building time, because they are assumed to be constant during a single platooning trip and are thus of sporadic nature. The value range and quantization declarations represent realistic numbers that are valid in the context of heavy weight trucks.

Although required and provided services have only been mentioned as black box interfaces of role configurations so far, the service hierarchy can only be built by also considering the inner function deployment within a role. An important aspect of service hierarchies is that services are not just abstract signal flows. Instead, a required service outsources a *responsibility* to another function or even configuration. This can happen either along or against the signal flow direction. In *Figure 4.15*, function *Safe Distance Computation* provides the abstract application service *Safe Platoon Driving* that is consumed by the human drivers of leader and follower truck. Thus, *Safe Distance Computation* has the responsibility to take care of the platooning collaboration being always safe during driving. Since complex systems are typically developed according to divide-and-conquer strategies, the systems are composed of communicating components or functions. In this way, *Safe Distance Computation* can only compute a safe distance set point out of given follower stop and leader brake distances. Hence, it cannot guarantee that the input distances have been computed correctly and could thus in theory compute an unsafe distance set point. Because of that, these two responsibilities are outsourced to the *Follower Stop and Leader Brake Distance Determination* functions. Note that in both mentioned cases, the service provision follows the same direction like the signal flow. This is different for the third and last responsibility that has been outsourced from *Safe Distance Computation* – the correct realization of a distance set-point. Whereas the function provides a safe distance set point as output, it could still happen that the set point is not actuated correctly turning the collaboration potentially unsafe. Thus, *Safe Distance Computation* demands from *Distance Realization* to realize its provided set point correctly. The responsibility outsourcing procedure is repeated until a responsibility can be completely satisfied by a single component or function. In the platooning system, these terminal components are the truck platforms of both roles, which fulfill their responsibility by providing data through sensors and actuation through the brake system and engine (only required for follower).

The selection of the function and therefore the role providing the application service is critical, because the responsibility for a safe overall collaboration should be only taken by a function which can decide if the collaboration's safe conditions are satisfied. *Safe Distance Computation* has the required context knowledge, since its functionality has been directly derived from the platooning safe condition defined in *Section 4.2*. *Distance Realization* or *Follower Stop Distance Determination* could not

decide solely based on their inputs and outputs, if the collaboration is currently safe or not and thus it does not make sense to assign the application service to one of these functions.



Service Type	Description	Unit	Range	Quantization	Minimum Frequency
get_V	The current longitudinal speed v of the vehicle (Continuous)	$\frac{m}{s}$	$0 \leq v \leq 40\frac{m}{s}$ $40\frac{m}{s} = 144\frac{km}{h}$	$0.01\frac{m}{s}$	10 Hz
get_sBrake	Get the truck's brake distance s_B in case it performs a full brake with maximum deceleration from the current vehicle state (Continuous)	m	$0 \leq s_B \leq 500m$	$0.1m$	$10\ Hz$
get_RoadInc	The inclination q of the road Remark: The maximum allowed road inclination on European highways is 8% (Continuous)	%	$0 \leq q \leq 10$	0.1%	$10\ Hz$
get_RoadFrict	Slip coefficient μ that describes how much of the applied de-/acceleration force can be transferred from tires to road (Continuous)	-	$0 \leq \mu \leq 1.2$	0.1	$10\ Hz$

Figure 4.16: Service architecture - Platooning scenario S2

The service architecture of scenario S2 is shown in *Figure 4.16*. The service interfaces of the role configurations differ from those of scenario S1 due to a different function deployment resulting from truck platforms providing different capabilities. It can be observed that scenario S2 yields only four provided/required inter-role services. Although the amount is significantly lower compared to scenario S1, the sporadic services that have been required from *Leader Brake Distance Determination* in case of its deployment to the follower role in S1, are replaced in S2 by the continuous service *get_sBrake* directly providing the leader's brake distance. In addition, *Follower Stop Distance Determination* requires two services *get_RoadFrict* and *get_RoadInc* from the leader role containing the road slip coefficient and the road inclination, respectively.

In summary, both service architectures provide a safe nominal behavior specification for platooning by the orchestration of different role configurations being able to provide the application service *Safe Platoon Driving* to both drivers. More specifically, the service architectures enable safe platoon driving provided that no systematic software or random hardware failures propagate up within the service hierarchy finally prohibiting the application service provision. Although both architectures yield a safe collaboration, they nevertheless differ in the following aspects:

Amount of services The amount of services between roles has a major impact on the flexibility that an OAS manufacturer has for the implementation of a certain role, because the provided services of a configuration dictate exactly, which quantity has to be provided at which accuracy. Hence, the more compositional the service types are in the end (e.g. *get_sBrake* vs. *get_RoadInc*), the lower the amount of services and the higher the flexibility for an OAS manufacturer will be.

Service impact on collaboration The exchange of information through services carrying continuous quantities has a much higher impact on the overall collaboration compared to sporadic services. This is not only the case because of strict timing requirements, but also with respect to a required continuous service provision, which is much more sensitive to failure occurrences due to higher service consumption frequency during the collaboration.

Functional performance The functional performance of the collaboration, i.e. the safe distance value during platoon driving, is dependent on the chosen configurations, more specifically on the assumed truck platform capabilities for the configurations.

Having engineered a safe nominal behavior specification for the platooning collaboration assuming the absence of failures, *Chapter 5* will examine how the platooning system behaves when systematic software or random hardware failures occur in both trucks. In order to guarantee a safe collaboration in this case, too, each role configuration will be eventually enriched with a ConSert introducing a variant restriction from the safety perspective.

5 Engineering safe fail behavior

Based on a safe nominal behavior specification in the shape of a defined service architecture, this chapter will first explain in *Section 5.1* how failures propagate through OAS collaborations and how they can be systematically analyzed and documented as safety property types. Next, *Section 5.2* presents a simulative approach for the quantification of deviation bounds for the safety properties of the collaboration interface. Subsequently, *Section 5.3* presents some general considerations of how an argumentation for the safety of the collaboration can be developed. Finally, ConSerts will be created for each OAS in *Section 5.4* mapping safety guarantees to safety demands and thus yielding runtime certificates that guarantee safety for the platooning collaboration. *Figure 5.1* shows the sequence of activities and their mapping to sections. Note that, due to space constraints, only the platooning scenario S2 (refer to its service architecture in *Figure 4.16*)) will be used from this point on for the illustration of the concepts described in this chapter.

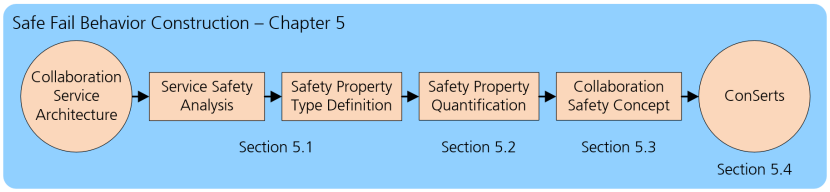


Figure 5.1: Safe fail behavior construction – Chapter structure

5.1 Service safety analysis

This section first explains general considerations for the safety analysis of OAS collaborations in *subsection 5.1.1* and subsequently applies a HAZOP analysis on the functional service types of the platooning system in *subsection 5.1.2*.

5.1.1 Safety analysis for OAS collaborations

Before safety properties and ConSerts can be defined in order to guarantee the provision of the application service in OAS collaborations, it is necessary to understand the differences in what way failures propagate through OAS collaborations compared to closed systems. To that end, *Figure 5.2* illustrates the situation for a collaboration of two OAS, where both

OAS realize a set of functions with the collaboration interface consisting of only one service.

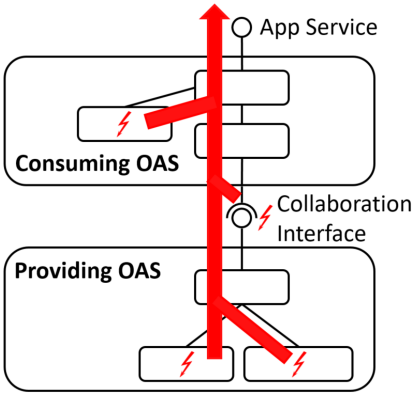


Figure 5.2: Failure propagation in OAS collaborations

The eventual goal of the safety assurance activities of this chapter is to guarantee that the application service's safety properties are always satisfied during platoon driving. Thus, it is of high importance to identify all potential failure modes of the collaboration that may prohibit the provision of the application service. In the same way as in closed systems, there are two types of failures that may also occur within a single OAS: random hardware failures originating from the OAS hardware platform and systematic software failures that either stem from the implementation of the application's functionality or from software providing infrastructure services like the operating system. The collaboration interface as such is an additional source of failures in OAS collaborations because of the underlying communication infrastructure introducing new potential failure modes. In *Figure 5.2*, the failure sources are indicated with red lightning bolts, while their propagation towards the application service is illustrated with a red arrow.

A fundamental difference to closed systems is where failures may manifest themselves and therefore which possibilities exist for their detection and handling. If we temporarily only examine the *providing OAS* and for instance consider a hazardous single bit flip in its RAM memory, it can be observed that this is a hazard, which can be easily mitigated by standard mechanisms like a parity check directly in the *providing OAS*. In this way, the *consuming OAS* never notices that a hazardous state existed. The situation is different for failures causing behavioral deviations that can only be perceived when examining the overall collaboration. The required context knowledge for this judgment is only available in the OAS providing the application service, which is the *consuming OAS* in Figure 5.2. Hence, the OAS providing the service cannot detect or mitigate the failure, i.e. the failure will inevitably manifest itself at the collaboration interface and thus propagate towards the *consuming OAS*. From the point of view of the

consuming OAS, a received faulty value cannot be distinguished from a correct value unless redundancy is in place, because the functional service types documented in the service architecture do not make any statement about potential deviations of service behaviors. Instead, they assume the service behavior to be always delivered correctly.

Due to this distinction inability, the only chance to guarantee a safe provision of the application service is to tolerate failures to a certain degree. In the platooning example, this means the inter-truck distance can be increased to compensate for potential failures, which leads to a worse functional performance for the sake of fault tolerance. In order to properly implement this fault tolerance mechanism, it is of essential importance to have quantified bounds on the deviations that may occur for a certain service. This is required to determine concrete control values that realize the failure compensation.

Before being able to specify deviation bounds for services, it is first of all necessary to analyze each functional service type separately for potential deviation types having safety-critical effects on the collaboration. In the following, the types of safety-critical service behavior deviations will be called safety property types according to the terminology of the ConSerts approach. As a starting point of the safety analysis for OAS collaborations, [17] used the hazard and operability study (HAZOP) on the functional service types. The causal model of HAZOP is compatible with the notion of failure manifestation described above, because by examining the potential effects of safety-critical deviations at the collaboration interface, it is of secondary importance, where exactly a failure occurred in the system. By using a top-down approach starting at the service-level, any combinations of failures are covered, too, because their propagation will eventually reach the collaboration interface and hence also have an effect on the service behavior. Thus, the identification of all potential safety-critical effects on the service behavior and their transformation into safety property types yields a safety interface that can enrich functional service types with safety information. Since the mere effects of a service behavior deviation do not have a relation to the deviation's causes, the reuse of functional service types together with their associated safety property types in different collaborations is possible. In addition, the modularity of safety property types and functional service types facilitates the enrichment with respect to other quality attributes like security in the future, too.

HAZOP guides the safety engineer by providing a set of guide words indicating typical failure modes which shall be taken into consideration during the analysis of a system. However, the set of applicable guide words is typically tailored to a specific application domain. With respect to OAS, it is even the case that different kinds of services have typical failure modes leading to a different interpretation of guide words for each kind. A first step in this direction has been proposed in [35] containing a classification of services (*Figure 5.3*) with respect to their usage purposes in OAS collaborations. Concerning the application of HAZOP, the service classifica-

tion can be used to tailor sets of guide words for the different mentioned classes. For instance, while the failure modes of a perception service like the exchange of a continuous speed value will mainly target value failures, an arbitration service like the realization of a control set point will have failure modes focusing rather on timing and provision failures. The long-term prospect with respect to these service classification efforts is a standardized domain-specific repository of basic functional service types being classified according to *Figure 5.3* including typical class-specific HAZOP guidewords and failure modes. In this way, a service could be easily reused together with its safety information as a complete package among different collaborations of a domain. This packaging of experience would not only improve the completeness and accuracy of failure modes due to the addition of more complex failure modes to the repository over time, but would also prevent safety engineers from repeating the analysis for similar services.

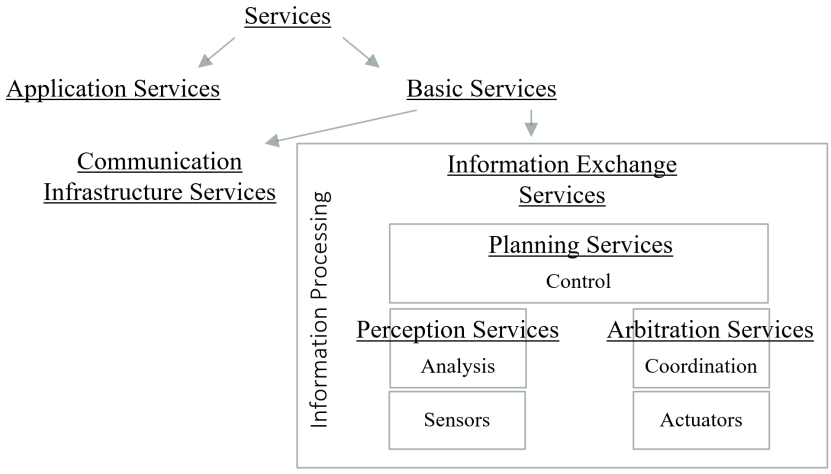


Figure 5.3: Service classification within OAS collaborations [35]

After the failure modes have been analyzed for the collaboration interfaces between all collaborating OAS, they shall be documented as safety property types associated with the respective functional service types including the following information:

1. A description of the potential deviation from correct service. This represents an interpretation of the HAZOP guideword for a given service.
2. The possible consequences that a deviation might have on the given collaboration. These consequences are critical for the assessment of the deviation's safety risk and therefore, they influence the respective safety properties integrity level (e.g. ASIL).
3. A description of potential causes that typically lead to the deviation. This item is optional, but if that information is already available at this

point it helps during a further deductive in-depth analysis of the deviation's causes either within the single OAS or in the communication infrastructure.

More information on the relation between functional service types, safety property types and their instantiation in shape of concrete services and safety properties can be found in *Section 2.2.2*, where the operationalization of ConSerts is described.

5.1.2 Platooning safety property definition

This section presents and explains the results of the safety analysis according to HAZOP for the platooning collaboration's service interface. As a basis for the HAZOP analysis, the following commonly known set of guide words has been used: omission, commission, too low, too high, too early, too late. A more detailed taxonomy of failure types and therefore an extended set of guide words has been lately proposed in [36], which, apart from provision, content and timing domains, explicitly distinguishes between sporadic and permanent failures, too. However, this differentiation did not yield additional findings for the analysis of the platooning system.

The identified safety property types for the service types *get_sBrake* and *get_V* (referring to the service architecture in *Figure 4.16*) are listed in *Figure 5.4*. Due to a lack of space, the safety property type description of the other services *get_RoadFrict* and *get_RoadInc* can be found in *Figure A.1* in the appendix of this thesis.

The approach for applying the guide words was to look at first only at the service type descriptions and interpret the guide words for the given services. This yielded the information documented in the first three columns of the table. The description of a service behavior deviation has been split into two parts, namely a qualitative description and quantitative refinement parameters, because a specific deviation might exist in different collaborations, but with different values for the parameters. This procedure also prepares the next step in the process, where the refinement parameters will be concretized for the given specific collaboration. The last two columns of the table consider a judgement whether a certain deviation is safety-critical for the given collaboration and gives indications for the causes of the failure mode.

Since all examined service types carry continuous signals of physical quantities, the first identified pattern has been that the guide words "commission", "too early" and "too late" do not make sense for these kinds of services. The underlying reason is that the benefit of a continuous "getter" service gets larger, the more frequently a value is delivered. Therefore, an unrequested delivery is not a failure mode in this scenario. In addition, failure types from the timing domain stemming from too early or too late are typically applied to event-based signals where a specific point in time is

Safety Property Type	Deviation from correct service	Refinement parameters	Potential causes	Possible consequences for collaboration
Commission, too early and too late are not meaningful for services delivering continuous signals				
Functional service type get_sBrake (Leader brake distance provision)				
Omission	No brake distance value is received at the service consumer for more than y time units	Omission time y ms	Truck out of comm. range, comm. medium overload	Safety-critical, because Safe Distance Computation lacks the required knowledge for computing a proper safe distance set point, thus an unsafe distance can arise. A switch to some worst-case value or immediate cancelling of the platoon are possible.
Too low	Received brake distance is more than x m lower than the actual value	Maximum deviation x m	Leader Brake Distance Determination failure, Environmental noise	Not safety-critical: A lower perceived leader brake distance yields a higher computed safe distance according to the platooning safe distance definition → distance increases and is thus safe
Too high	Received brake distance is more than x m higher than the actual value	Maximum deviation x m		Safety-critical: A higher perceived leader brake distance yields a lower computed safe distance according to the platooning safe distance definition → distance decreases and might be unsafe
Functional service type get_V (Leader speed provision)				
Omission	No speed value is received at the service consumer for more than y time units	Omission time y ms	Truck out of comm. range, comm. medium overload, Speed sensor failure	Temporarily safety-critical, until a switch to the redundantly provided speed value by follower's front radar is done. The interval until the detection might yield a safety-critical distance, if the leader truck's speed decreased during that time.
Too low	Received speed is more than x $\frac{km}{h}$ lower than the actual value	Maximum deviation x $\frac{km}{h}$	Speed sensor failure (drift, bias, etc.), Rounding errors in Leader Platooning SW, Environmental noise	Not safety-critical: Follower controls v_relative = 0, so a too low leader speed value yields a realization of a higher (=safe) distance set point for compensation of speed error
Too high	Received speed is more than x $\frac{km}{h}$ higher than the actual value	Maximum deviation x $\frac{km}{h}$		Safety-critical: Follower controls v_relative = 0, so a too low leader speed value yields a realization of a lower (=potentially unsafe) distance set point for compensation of speed error

Figure 5.4: Safety property types of service *get_sBrake* and *get_V*

required for a signal. These failure modes are more relevant when dealing with arbitration services, where other OAS should be remotely controlled through a service. The examined services are perception services only and thus timing failures are irrelevant.

Hence, perception services are only prone to two types of failure modes: Firstly, an omission, i.e. an absence of service provision, and in case the service is provided, the values can be either lower or higher than the actual value. An interesting finding in this respect is that with all examined service types, only a value deviation in one direction was judged as being safety-critical. This is a logical consequence, when the effects on the collaboration are studied, because all perception services influence the computed safe distance set point. Thus, when the correct value of a perception service yields exactly the correct safe distance, a deviation will either result in a too short or too long distance set point of which only the short one is safety-critical for the collaboration.

Having determined the primary failure modes of the basic functional service types of the service interface among follower and leader, a special consideration is needed for the application service *Safe Platoon Driving* being provided by the follower truck. Since the application service is formulated on a very abstract level, the HAZOP guide words cannot be applied to it, because whether the platoon driving can be guaranteed for all participants in a safe way or not is a binary decision only. The most suitable failure type in this respect would be an omission of the application service with an omission time of 0 ms to emphasize that the service must be always provided during platoon driving. At first glance, the annotation of the omission safety property type seems superfluous, but if the eventual definition of ConSerts is considered, the application service's safety properties will be the provided safety interface of the ConSert for the platooning system of the follower role.

In the agricultural case study in [5], the safety property types have been additionally annotated with a situation-related refinement, i.e. whether a certain failure mode has different effects and therefore different criticality depending on the situation. This additional information was useful in the TIM scenario, because a distinction between the very different situations standstill and normal operation has been used and therefore yielded additional input for the risk assessment. However, platooning as defined in this thesis considers a platoon to be driving on highway with high speeds as the only "major" situation. Therefore, an additional situation-related analysis was not performed as part of this thesis.

5.2 Simulative safety property quantification

This section presents a simulative approach for the quantification of refinement parameters for the safety property types, which have been identified for the functional service types of the platooning system in the previous section. To that end, *Section 5.2.1* will describe the general concept and the underlying considerations for the choice of numerical simulation as a means for the quantification. Afterwards, building on top of the qualitative functional decomposition of *Section 4.2*, a detailed physical model of platooning suitable for simulation will be given in *Section 5.2.2*. The realization of the simulation model in Matlab/Simulink™ as well as the simulation procedure will be illustrated in *Section 5.2.3*. Finally, the safety property refinements based on the simulation results will be given in *Section 5.2.4*.

5.2.1 Quantification concept

The quantitative refinement of safety property types with respect to a concrete collaboration scenario is a core activity for the construction of a safe fail behavior specification with ConSerts. Considering the anatomy of a

ConSert, it maps safety guarantees for provided services to safety demands of required services of a collaboration role configuration, during which guarantees as well as demands are expressed through safety properties. The argumentation of a ConSert for guaranteeing a safe behavior follows the idea that a safety guarantee can only be given, if all safety demands being mapped to that guarantee are satisfied by other collaboration partners. These demands require that the safety properties of the required services are not violated, i.e. that the identified deviations will always stay between fixed maximum boundaries. A quantification of the maximum boundaries for all safety properties is necessary, because otherwise there would be no means to reason about the overall collaboration's safety.

As soon as fixed deviation boundaries are specified for each service, these boundaries can be used to actively compute set points for controlling the collaboration state so that deviations within these boundaries can be tolerated in a safe way. The variable for controlling the state of the platooning collaboration is the set point for the inter-truck distance. The idea is to increase this distance when big ranges of deviations should be tolerable or to decrease the distance in case only small deviations can be guaranteed by the truck providing the services. Obviously, when increasing the distance, the air drag is also increased for the follower truck and thus the benefits in terms of saved fuel decrease. Hence, an optimal functional performance is only possible, if minimal deviation ranges can be guaranteed.

For the following considerations, potential service behavior deviations have to be integrated into the safe distance definition given in *Section 4.1.2*. In order to determine, how much additional separation between the trucks is needed for safely tolerating specified deviations, a new physical quantity called safety margin d_{margin} shall be introduced. For this purpose, *Figure 5.5* extends the visualization of the safe distance during failure absence (see *Figure 4.3*) with the quantities $\Delta s_{brake,L}$,$\Delta s_{stop,F}$ and d_{margin} stemming from safety-critical deviations.

The upper part shows the situation where a crash will occur after an emergency braking maneuver, because both $s_{stop,F}$ and $s_{brake,L}$ are increased due to deviations. Note that the increase amount $\Delta s_{stop,F}$ is much higher than $\Delta s_{brake,L}$. The additional distance that needs to be maintained *before* a braking maneuver to compensate for these deviations is the safety margin d_{margin} indicated with the red arrow in both situations. Based on this observation, the **safety margin** can be formulated as:

$$
\begin{aligned}
& d_{safe} + d_{margin} + s_{brake,L} + \Delta s_{brake,L} = s_{stop,F} + \Delta s_{stop,F} \\
& \quad \Longrightarrow \; d_{margin} = \Delta s_{stop,F} - \Delta s_{brake,L} && \text{Functional level} \\
& \quad \Longrightarrow \; d_{margin} = f(\Delta s_1, \Delta s_2, \cdots, \Delta s_n) && \text{Service level}
\end{aligned}
$$

with Δs_i : value deviation of required service s_i

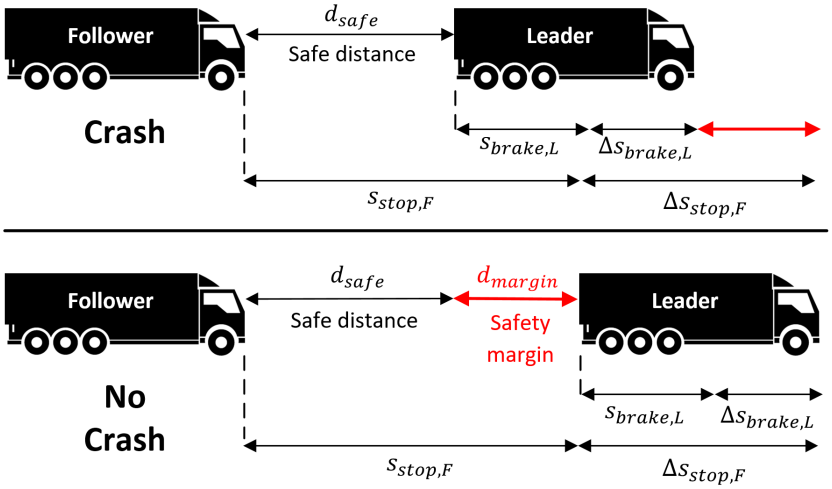


Figure 5.5: Impact of service behavior deviations on collaboration safety

The service level definition above shows that d_{margin} is defined by a function of deviations of scenario-specific required services, which eventually influence $\Delta s_{stop,f}$ and $\Delta s_{brake,L}$. For the quantification of the safety properties, this means there are two possible procedures for the determination of maximum deviation bounds yielding a safe collaboration:

1. d_{margin} is fixed to a certain value and all combinations of Δs_i are searched for so that the above service-level equation holds. Descriptively spoken, this means a specific inter-truck distance should be fixed guaranteeing a certain functional performance. The optimization goal is to find sets of maximum value deviation bounds for all services so that the required functional performance can be achieved.
2. Specific Δs_i are selected as fixed and the required additional distance d_{margin} is computed. The descriptive goal of this procedure is to have a set of maximum deviation bounds fixed, e.g. based on an existing platooning realization, and compute the best functional performance that can be achieved with this setting. The results can be used to reason about the effects of larger or smaller allowed deviation ranges on the functional performance.

Both described quantification procedures can be carried out either analytically or with a simulative approach. The analytic variant for the platooning system would consist of a mathematical refinement of the functional-level safety margin definition with respect to the influence of given services on $\Delta s_{stop,F}$ and $\Delta s_{brake,L}$ in a specific scenario. However, considering the complexity of realistic systems, it is hardly possible with reasonable effort to analytically determine the function f of the service-level definition above.

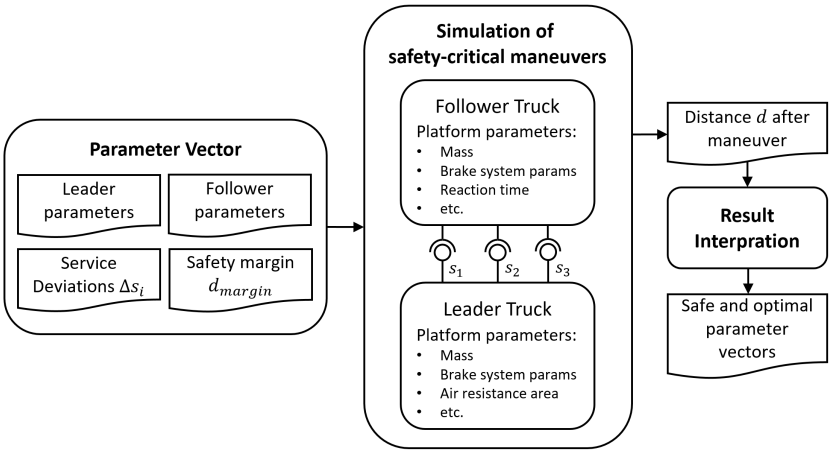


Figure 5.6: Simulative quantification of safety properties

Thus, we selected the simulative variant in combination with the first quantification procedure delineated above and created an executable simulation model for the platooning system. For a simulation, both leader and follower roles are treated as black boxes and different combinations of service value deviations can be realized by failure injections at the service-level. In this way, the parameter vector for one simulation run of an emergency braking maneuver contains on the one hand a set of fixed service deviations and on the other hand a fixed value for d_{margin}. The simulation has to be performed for a pre-defined set of parameter vectors. Eventually, a parameter vector can be considered *safe* for a simulation run, if the simulation's braking maneuver does not yield a crash among both trucks. In addition, a parameter vector is *optimal*, if the inter-truck distance after the brake maneuver is minimal, but still positive. The simulation procedure for the quantification of safety property refinements is illustrated for the platooning system in *Figure 5.6*.

Based on the simulation results, different sets of service deviations can be chosen that yield a safe and preferably optimal collaboration with respect to the functional performance. The chosen deviation sets are eventually used for the definition of the deviation boundaries within the safety property refinements with respect to specific collaboration scenarios.

5.2.2 Physical model building of a truck

The chosen simulative approach for the quantification of safety properties requires to build a simulation model of the collaboration. Given the simulation approach in *Figure 5.6*, the requirements for the platooning simulation model are:

1. Create a physical model of platooning considering all relevant characteristics of both trucks as well as the collaboration environment, i.e.

the behavior of all inner functions of the functional model defined in *Section 4.2.2* have to be concretely modeled.

2. Provide a framework enabling a batch simulation of a defined set of parameter vectors and data recording for the simulation results.

While the remainder of this section will focus on the creation of a realistic physical model for platooning, the realization of its batch simulation in Matlab/Simulink™ will be the content of *Section 5.2.3*.

Physical motion model of a truck

In order to build a physical model of platooning that can finally be simulated, an exact mathematical representation is required for platooning in general and for the considered scenarios in particular. The essential safety-critical scenario, which shall be simulated, is the emergency braking maneuver starting in the platoon's steady state, i.e. where both trucks are platooning with the same speed v_0 and an initial distance d_0. The brake scenario is initiated by the leader driver, who brakes with maximum brake force until the leader truck comes to a stop. The follower truck autonomously reacts with full braking to stop, too, as soon as leader braking is perceived.

According to the functional decomposition in *Section 4.2*, both leader brake distance and follower stop distance are needed for the computation of a safe distance and eventually a deceleration set point for the follower truck. This requires a physical model of each truck expressing how braking affects the truck's motion state, i.e. the acceleration $a(t)$, speed $v(t)$ and traveled distance $s(t)$, where the speed is obtained by single integration of $a(t)$ over time, whereas traveled distance results from two-time integration of $a(t)$ over time. Note that we are finally interested in the traveled distance, because its value at standstill is equal to the brake/stop distance. The temporal braking process for each truck as such will be modeled according to the one described in *Figure 4.6* in *Section 4.2*. As already indicated there, a closed form solution for the stop distance of the braking process is available in [31], but this solution does not consider truck characteristics and influences, which are of importance for the platooning collaboration defined in this thesis such as the air drag having a major impact on a realistic computation of the stop distance.

In the following, the motion state of a truck should be mathematically derived for a given *requested* deceleration $a_{set}(t)$ under the influence of air, rolling, acceleration and inclination resistances. Therefore, the first step is to create a force diagram of all forces, which act on a truck during braking (*Figure 5.7*).

While the forces F_{Air}, F_{Roll}, F_{Road} and F_{Weight} stem from specific truck characteristics and the environment, F_{Brake} is the force that the brake system transfers through the wheels on the street according to the request of the driver or, in the autonomous case, of the system. $F_{Vehicle}$ represents

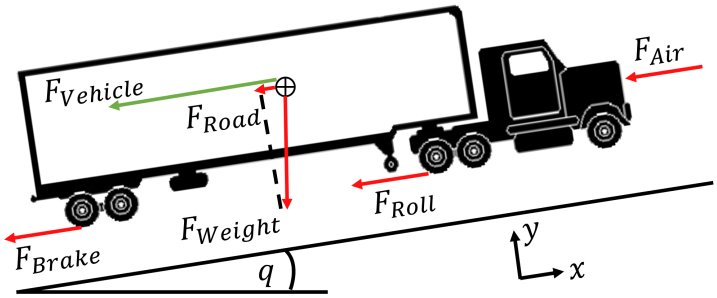


Figure 5.7: Driving resistance forces acting on a truck

the resulting force that is finally affecting the truck's motion. Note that the inclination resistance F_{Road} is a consequence of the truck's weight force F_{Weight} and its value is that component of F_{Weight} acting in parallel to the road. Thus, F_{Weight} does not need to be considered in the following. The sum of all forces with correct signs yields the resulting force $F_{Vehicle}$:

$$F_{Vehicle} = -F_{Brake} - F_{Air} - F_{Road} - F_{Roll} \tag{5.1}$$

According to the standard literature, e.g. [31, p. 38-96], the resistance forces can be replaced with a mathematical relation of their influential quantities.

$$\lambda_m m \cdot a(t) = -m a_{set}(t) - \frac{1}{2}\rho c_w A \cdot v^2(t) - mg\frac{q}{100} - f_r mg \tag{5.2}$$

with λ_m = rotational mass factor, m = truck mass,
$a(t)$ = truck de-/acceleration, $a_{set}(t)$ = requested de-/acceleration,
ρ = air density, $c_w A$ = air resistance area, $v(t)$ = truck speed,
g = gravity, q = road inclination, f_r = roll resistance coefficients

The consideration of $a(t) = v'(t)$ and the subsequent isolation of $v'(t)$ in **(5.2)** leads to **(5.3)**, where all time-independent coefficients and terms can be summarized into single constants **(5.4)**.

$$v'(t) = -\frac{1}{\lambda_m} a_{set}(t) - \frac{\rho c_w A}{2\lambda_m m} \cdot v^2(t) - \frac{g \cdot q}{100\lambda_m} - \frac{f_r g}{\lambda_m} \tag{5.3}$$

$$= -\frac{1}{\lambda_m} a_{set}(t) + K_{Air} \cdot v^2(t) + K_{Road} \tag{5.4}$$

with $K_{Air} = -\frac{\rho c_w A}{2\lambda_m m}$ and $K_{Road} = -\frac{g \cdot q}{100\lambda_m} - \frac{f_r g}{\lambda_m}$

Equation **(5.4)** finally represents the 1st order, 2nd degree inhomogeneous differential equation (DE) with constant coefficients describing the motion of a truck, where the driver's requested de-/acceleration $a_{set}(t)$ is the stimulation of the truck system. With the computation of a DE solution for $v(t)$, both $a(t)$ and $s(t)$ can be determined by derivation and integration, respectively.

Stop distance computation

This section instantiates the previously defined motion model of a truck for the different phases of a braking process. The braking process has been introduced already in *Section 4.2.2*, so *Figure 5.8* shows only the function course for the requested deceleration $a_{set}(t)$ together with a partition of the stop distance s_{stop} into the sections s_{react}, $s_{b,dwell}$ and $s_{b,max}$ belonging to the different braking phases.

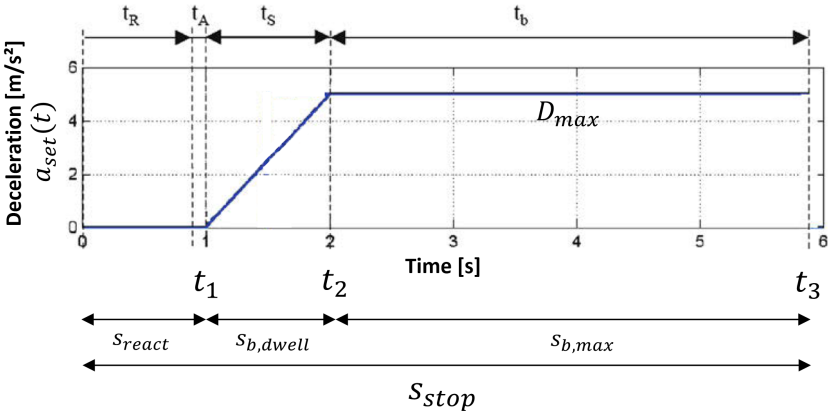


Figure 5.8: Stop distance composition for the braking process

The reason for partitioning s_{stop} into three sections is that the course of $a_{set}(t)$ can be also divided in three sections having a homogeneous shape each.

$$a_{set}(t) = \begin{cases} 0 & 0 \leq t \leq t_1 \\ \frac{D_{max}}{t_S} \cdot t & t_1 < t < t_2 \\ D_{max} & t_2 \leq t \leq t_3 \end{cases} \tag{5.5}$$

Both the maximum deceleration D_{max} and the dwell time t_S are known constants and thus, the truck motion DE **(5.4)** can be formulated for the braking process as:

$$v'(t) = \begin{cases} K_{Air} \cdot v^2(t) + K_{Road} & 0 \leq t \leq t_1 \quad \textbf{(5.6)} \\ K_{Air} \cdot v^2(t) + K_{D_1} \cdot t + K_{Road} & t_1 < t < t_2 \quad \textbf{(5.7)} \\ K_{Air} \cdot v^2(t) + K_{D_2} + K_{Road} & t_2 \leq t \leq t_3 \quad \textbf{(5.8)} \end{cases}$$

with $K_{D_1} = -\frac{D_{max}}{\lambda_m t_S}$ and $K_{D_2} = -\frac{D_{max}}{\lambda_m}$

While equations **(5.6)** and **(5.8)** are structurally equal, **(5.7)** includes an addition linear term being dependent on t. Thus, in order to compute explicit solutions for the traveled distance $s(t)$ in all phases, two types of differential equations need to be solved considering the relation $s(t) = \int v(t)$

as well as the boundary conditions $s(0) = 0$, $v(0) = v_0$ and $v(t_3) = 0$ for the braking process. In addition, the value continuity of $v(t)$ at t_1 and t_2 needs to be guaranteed among the involved solutions at these times, i.e. the initial speed of the phases starting at t_1 and t_2 must be equal to the end speeds of their previous phases. Due to the lack of space, background information on the DE types as well as hints to their mathematical solution procedures are given in the appendix section A.1.

Having explicit functions for the traveled distances $s_1(t)$, $s_2(t)$ and $s_3(t)$, the stop distance s_{stop} can be determined as:

$$\begin{aligned} s_{stop} &= s_{react} + s_{b,dwell} + s_{b,max} \\ &= s_1(t_1) + s_2(t_2) + s_3(t_3) \\ &= s_1(t_R + t_A) + s_2(t_S) + s_3(t_B) \end{aligned} \tag{5.9}$$

Distance and speed control

With respect to the functional model of platooning given in *Section 4.2.2*, the only function for which no explicit behavior specification exists so far, is the controller that transforms set points for truck distance and relative speed into a de-/acceleration set point for the follower truck. For the simulation model, a controller design from [32, p. 872ff.] has been chosen that is generally applicable for adaptive cruise control systems (ACC). Its transfer function is given in equation **(5.10)**.

$$a_{i+1}(t) = \left(v_{rel} - \frac{d_{set} - d_{actual}}{\tau_d} \right) / \tau_v \tag{5.10}$$

The controller primarily controls the relative speed of the trucks to be $v_{rel} = 0$. The distance error $d_{set} - d_{actual}$ is interpreted as an error of v_{rel} and is thus subtracted from it. τ_d and τ_v are parameters for the controller allowing a fine tuning of how aggressive control differences are controlled, i.e. how high the resulting de-/accelerations are. This is an important issue with respect to the physical limitations for the realization of de-/accelerations, especially in the case of trucks.

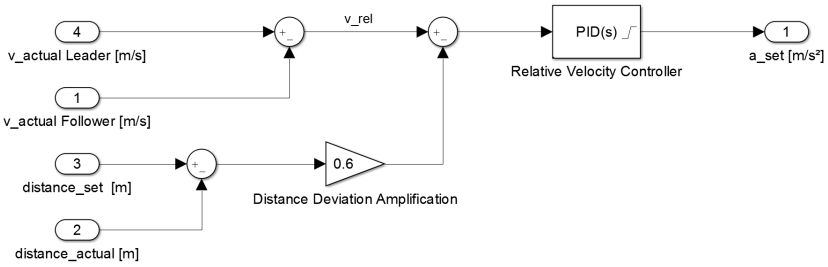


Figure 5.9: Block diagram of distance and speed controller

Figure 5.9 shows a block diagram of the controller being equivalent to equation **(5.10)**. τ_d is represented with the "Distance Deviation Amplification", while τ_v is hidden as the PID controller's proportional element.

5.2.3 Realization in Matlab/Simulink

This section shall give a short overview of how both simulation model and procedure have been realized in the popular behavioral modeling tool Matlab/Simulink™ of the *MathWorks* company. In general, it can be stated that a simulation model of an embedded system in the automotive domain has to consider three aspects:

1. a model of the system under development (the platooning system)
2. a model of the environment acting as source for sensing activities and as sink for system outputs affecting the environment
3. a model of the (human) driver(s) operating the system

All of these aspects have been modeled in Simulink for the platooning system. An overview of the model is presented due to space constraints in *Figure A.2* in the appendix. Note that both parts of the figure have to be imagined to be horizontally side-by-side to get the complete overview. In order to control the simulation of the Simulink model, a graphical user interface (GUI) has been developed (*Figure 5.10*) that enables several simulation possibilities for the engineer.

There are two modes how the model can be stimulated, i.e. how the leader truck can be "driven": Firstly, the engineer can perform the simulation interactively in real-time by using the throttle at the top for acceleration and braking. This also includes a simple cruise control, which keeps the speed constant if activated, and a button performing an instant emergency braking maneuver. Secondly, pre-defined acceleration profiles can be created and simulated faster in simulation time. This enables a repeatable behavior for batch simulations, which is required for the safety property quantification. The remaining sections of the GUI allow a parameterization of the model with respect to the environment, the leader and follower trucks as service deviations. In the interactive mode, all of these parameters can be changed during simulation to examine their effects immediately.

For the batch simulation of the platooning system's behavior during the occurrence of service deviations, a Matlab script has been created that allows the variation of specified parameter ranges and the subsequent parallel simulation on computation clusters. This simulation method is called *Rapid Accelerator Mode* [37] and compiles the Simulink model into a native standalone executable, which enables the best achievable simulation performance that is technically possible with Matlab/Simulink.

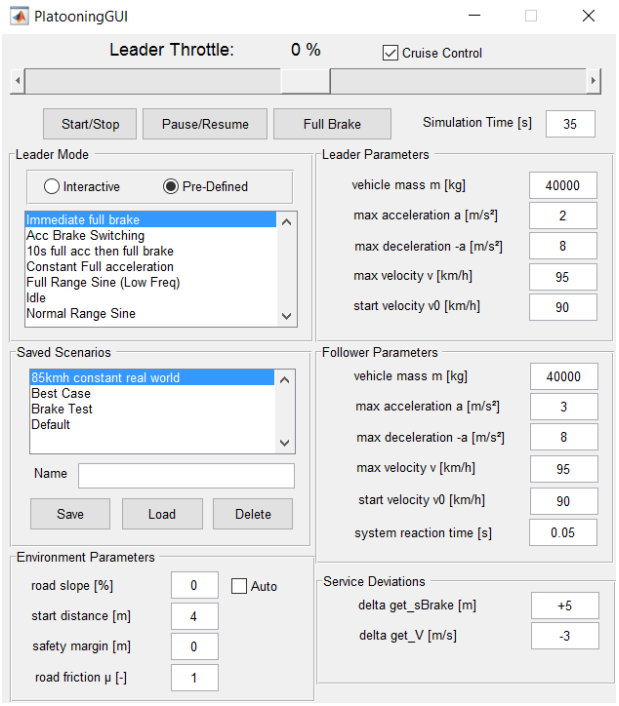


Figure 5.10: GUI for controlling simulations of the platooning system

5.2.4 Simulation results

Having explained how the simulation model of the platooning system has been created and how the simulation has been carried out with Matlab/Simulink, this section first presents the simulation results of concrete parameter vectors that have been simulated. Afterwards, it will be illustrated, how the simulation results lead to concrete safety property refinements.

The simulation was performed based on the platooning service architecture of collaboration scenario S2 (*Figure 4.16*). Although the batch simulation was optimized for Simulink's rapid accelerator mode and executed on the high-performance computation cluster of TU Kaiserslautern (16 cores, 128GB RAM), it was observable quite fast that the service deviation variation was not possible for *all* services of the scenario with reasonable step sizes for the variations. Therefore, we restricted ourselves to a variation of value deviations for the services *get_sBrake* and *get_V* providing the leader's brake distance and current speed, respectively. In addition, the safety margin d_{margin} has been varied. The truck platform parameters have been chosen to be equal for both trucks, more specifically, standard values for modern trucks have been selected.

Variation subject	Range	Step size	#Variations
Δv_L	[-10,+10] km/h	0.5 km/h	41
$\Delta s_{brake,L}$	[-10,+10] m	0.5 m	41
d_{margin}	[0,+12] m	4 m	4

Table 5.1: Simulation parameter variation

The chosen ranges, step sizes and the resulting number of variations per subject are given in *Table 5.1*. The combination of all variations with each other requires $41 \cdot 41 \cdot 4 = 6724$ simulation runs overall. The fact that the inclusion of variations of the scenario's missing services *get_RoadFrict* and *get_RoadInc* with a comparable variation granularity would yield $6724 \cdot 41 \cdot 41 = 11\,303\,044$ required simulation runs, suggests that the choice of a reasonable variation granularity is of high importance for avoiding scalability issues.

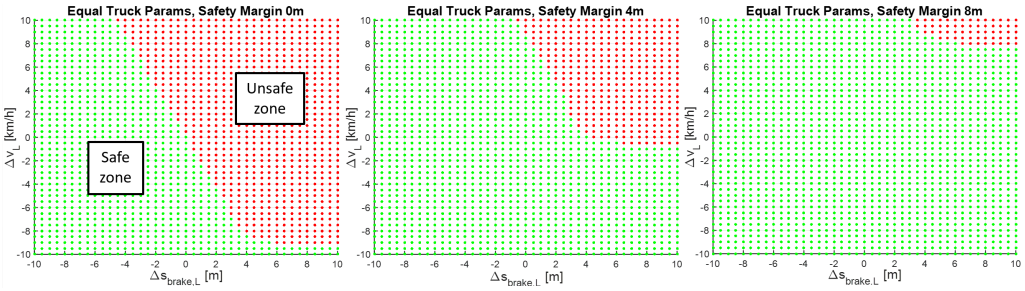


Figure 5.11: Simulation results

Figure 5.11 shows a comparison of the simulation results for three different safety margins of 0, 4 and 8 meters. The axes represent the amounts of value deviation for the two considered services, i.e. a data point is the result of a simulation run with a combination of these concrete deviations. The third dimension, which is presented as the color of the data points, represents the distance among the trucks being left after the braking maneuver. If a crash occurred for a certain simulation run, the left distance was negative, which is indicated with a red data point meaning that the combination of service deviations was unsafe. In the opposite case, where a positive distance was left after the maneuver, the data point is colored green suggesting a safe deviation combination. Thus, all data points that lie on the line separating safe from unsafe zone, belong to simulation runs that yielded an optimal functional performance, because the left distance after the maneuver was exactly zero in these cases, i.e. no unnecessary large distance was maintained *before* the braking maneuver.

Back in *Section 5.2.1*, the safety margin has been introduced as a means to compensate for increased deviations. This statement is supported by the plots of *Figure 5.11*, since the space of safe deviation combinations

grows all the larger, the higher the safety margin is chosen. Another aspect that could be verified through the simulation is the correctness of the safety property types that have been derived in *Section 5.1.2*. For both services *get_V* and *get_sBrake*, only the failure modes "Value too high" ($deviation > 0$) have been deemed safety-critical, which is acknowledged by the plots.

The process of how specific ranges of tolerable deviations can be determined for the services shall be explained in the following on the basis of *Figure 5.12*, where an annotated version of the simulation results for $d_{margin} = 4m$ is given.

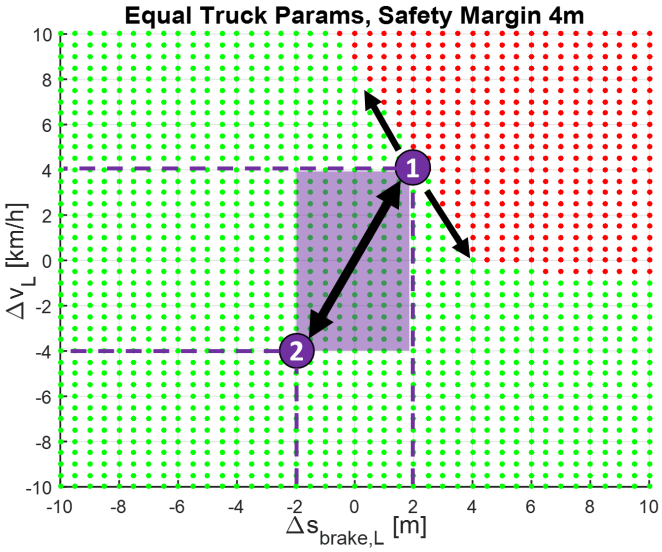


Figure 5.12: Selection of concrete safety property refinements

Because of the fact that the functional performance is optimal at the border line between safe and unsafe zone, it makes sense to select a point for the upper deviation bounds from that line (*Annotation 1*). Specifically for the illustration, the upper bounds $\Delta v_{L,max} = +4km/h$ and $\Delta s_{brake,L,max} = +2m$ have been chosen, i.e. if both services' values deviate exactly in this manner, a crash can be avoided just in time leading to an inter-truck distance of zero after the braking maneuver, if an additional distance of 4m (safety margin) has been maintained before the braking maneuver.

The next step is to choose corresponding lower bounds for the service deviations. Since the deviations are assumed to occur with equal probabilities in positive and negative directions, the lower bounds are determined as $\Delta v_{L,min} = -4km/h$ and $\Delta s_{brake,L,min} = -2m$ (*Annotation 2*). This observation explains why it is not possible to tolerate any deviations in the left plot ($d_{margin} = 0m$) of *Figure 5.11*. The reason is that all points lying

on the optimal line do not represent deviation sets that have only positive deviations and can therefore not serve as upper bounds.

Having selected both upper and lower bounds, the space of tolerable deviation sets is completely specified and is visually shown with the purple rectangle in *Figure 5.12*. Note that the lower bounds are not necessarily required from a safety perspective, because only the positive deviations ("value too high") have been deemed safety-critical in the corresponding safety property types. However, a determination of the lower deviation bounds guarantees a boundary for the maximum performance loss. This stands to reason, because the additional safety margin of 4 meters will be *always* maintained to guarantee a safe collaboration during value deviations, no matter if these deviations exist or not. In the case of negative deviations, the distance could even be chosen closer than in the case, where no deviations occur at all. As indicated in *Figure 5.12*, the maximum possible performance loss with the selected lower and upper bounds is 8 meters, which is exactly twice the amount of the chosen safety margin. Concretely, the maximum loss occurs in the situation, where $\Delta v_L = -4km/h$ and $\Delta s_{brake,L} = -2m$, because in this case the platoon could still collaborate safely if the truck separation would be 8 meters less.

Coming back to the safety properties being needed for the derivation of ConSerts, the idea is now to select one or more boundary sets according to the described process for the refinement of each safety property. For the running example, we chose two different safety property refinements that both guarantee a safe collaboration, but yield different functional performances. Note that, indeed, the functional performance has been used as an optimization criterion here, but there are also other conceivable optimization drivers, for instance the cost for guaranteeing that a service is provided with certain deviation bounds. Finally, the exemplary safety property refinements of the running example's collaboration scenario S2 are presented in *Figure 5.13*.

5.3 Collaboration safety concept

Having defined functional service types, service architecture and refined safety properties for all services of the collaboration interface, the next step is to provide a safety concept that can bring forward the argument that the collaboration is safe, i.e. that the following safety goal is met:

Safety Goal The platooning system must not cause a frontal crash between follower and leader trucks during driving on highway.

In order to achieve completeness, this safety goal has to be met in all three different phases of platooning, namely platoon building, driving and dissolving. The running example only considered the platoon driving phase in detail, thus only some general thoughts will be given for the other phases. The following elaborations assume that the platooning collaboration is

Functional Service Type	Safety Property Type	Refinement parameters	Concrete refinements	
General			Refinement 1	Refinement 2
get_sBrake	Too low	Maximum deviation x m	$\Delta s_{brake,min,1} = -2m$	$\Delta s_{brake,min,2} = -6m$
	Too high	Maximum deviation x m	$\Delta s_{brake,max,1} = +2m$	$\Delta s_{brake,max,2} = +6m$
get_V	Too low	Maximum deviation x $\frac{km}{h}$	$\Delta v_{min,1} = -4\frac{km}{h}$	$\Delta v_{min,2} = -8\frac{km}{h}$
	Too high	Maximum deviation x $\frac{km}{h}$	$\Delta v_{max,1} = +4\frac{km}{h}$	$\Delta v_{max,2} = +8\frac{km}{h}$
			Guaranteed maximum performance loss: 8m with $d_{margin} = 4m$	Guaranteed maximum performance loss: 16m with $d_{margin} = 8m$

Figure 5.13: Safety property refinements – Platooning scenario S2

safe in any of the phases, if the leader truck can perform an emergency braking maneuver without causing a crash between the trucks. The reason for this assumption is that other road users besides the two trucks as well as lateral movements have been explicitly excluded in the examined scenario.

Considering the building phase of the platoon, where both trucks are still driven by human drivers at first, it is required to check that the inter-truck distance is safe enough *before* the platoon is built. Note that the term "safe enough" is targeting the capabilities of the platooning system instead of the follower truck's driver, because the system has a much faster reaction time than a human and therefore the safe distance can be smaller if the system controls the follower truck. In contrast, platoon dissolving is triggered either by the intention of the drivers or because the system cannot further guarantee safe platooning. Of these two triggers, only the second one is highly safety-critical, because handing over the control back from the system to the driver is not instantly possible due to their different reaction times. Thus, the recognition that a safe collaboration cannot be guaranteed anymore must occur sufficiently early to ensure a timely increase of the truck separation. Taking into account that the follower truck driver might additionally not completely focus on traffic during the autonomous operation, [38] proposes a minimal time span of 10 seconds until a driver is ready to actively control his vehicle again. With typical truck speeds of 90 km/h, this means that the platooning system must be able to predict the collaboration's safety at least 250 meters ahead of the platoon. Given the dynamics of the platoon environment, this judgment is a tough challenge from a safety perspective.

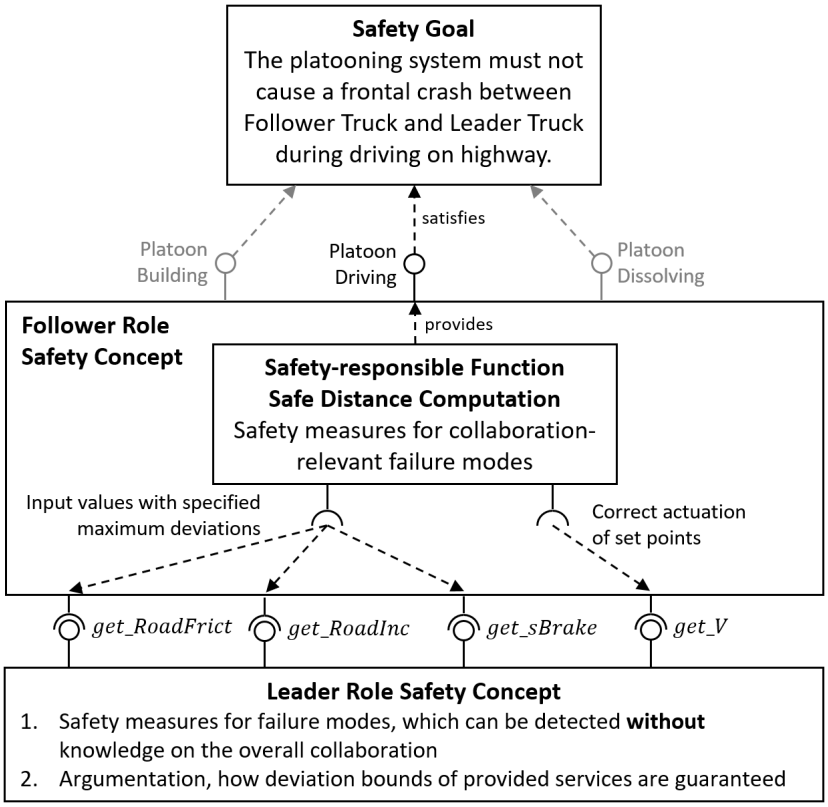


Figure 5.14: Abstract safety concept for the platooning system

In general, a safety concept for an OAS collaboration is supposed to provide an argumentation on two different abstraction levels (*Figure 5.14*): Firstly, an argument has to be presented for each collaboration role separately, showing how the safety properties of its provided services are guaranteed under the assumption that the safety properties of its required services are guaranteed from other roles. If a role's service interface and its safety properties are known in advance, this task can be carried out in isolation with the scope set to a single OAS only. In addition to this so-called role safety concept, it is necessary to have a dedicated role, which takes care of safety measures targeting the overall collaboration and therefore enabling the provision of the associated application service. This dedicated role has to be designed in a way so that it has enough context knowledge to determine the current safety status of the collaboration. Already during the function deployment activity in *Section 4.4.2*, the pursuit of having a role with high context knowledge became apparent. The underlying train of thought was that a high context knowledge enables taking the responsibility for guaranteeing the collaboration's safety, too. Within the platooning system, the function *Safe Distance Computation* provides the application service and is therefore also the so-called safety-responsible

function. Its required services are of two general kinds. On the one hand, it requires the provision of input quantities that have specified maximum deviation bounds and on the other hand, a correct actuation of its provided set points is demanded.

With respect to the required inputs, it is initially irrelevant for the safety concept as such, whether the values are provided from the follower truck platform or through services from the leader role provided that their deviation bounds are specified. Instead, the main task is to define safety measures as part of the safety-responsible function that describe how the identified failure modes like an omission or a value deviation of an input value are handled to guarantee a safe collaboration. In general, this includes the choice of fault tolerance mechanisms being able to detect and handle failures. If a failure cannot be detected or recovered from, a safe collaboration is potentially not possible anymore and thus, a transfer into a safe state is required. Considering the ConSert approach, the refinement of the collaboration interface's safety properties can be thought of as a fault tolerance mechanism, where the detection of failures is shifted to the design time and a safe handling of their potential effects is realized by specifying maximum deviation bounds by construction. Nevertheless, the safety concept has to include an argumentation, why a certain set of tolerated deviations for the required services yields a safe collaboration at all. In this respect, the approach within this thesis has been the simulation of deviations, but there might be other approaches for the argumentation, too.

So far, only safety properties describing *value* deviations of services have been safeguarded by requiring maximum deviation bounds. However, when it comes to permanent service omissions that for instance stem from a permanent sensor failure, the safety-responsible function needs to finally transfer the system into a safe state. If essential information for guaranteeing safe platooning is not present, one potential maneuver could be an immediate emergency braking maneuver of the follower truck to definitely prevent a frontal crash with the leader truck. Although this concept would satisfy the safety goal in the running example, it would probably cause accidents in real-world scenarios with road users driving behind the follower truck, because they do not expect sudden emergency braking and may not maintain the statutory separation to the follower truck. Another alternative in this scenario could be pro-active braking with moderate deceleration only following the argumentation that the probability of occurrence of a service omission and an emergency braking maneuver of the leader truck at the same time is acceptably low. A more general approach for safe-guarding omissions is the installment of redundancy patterns realized in hardware or software. For instance, if a permanent sensor loss is considered leading to an omission of a service, providing a second one of the same kind can be a solution.

In summary, this section described that compliance with the collaboration's safety goal can be split in two parts. Firstly, the realization of collaboration-

level safety measures should be concentrated on one function within the role providing the application service and having the highest context knowledge. This so-called "safety-responsible" function is the only entity within the collaboration that is able to both detect and thus safeguard failures being critical for the overall collaboration. In contrast, failure modes which do not propagate across the collaboration interface or which can be detected without considering the collaboration, can be safeguarded as part of the role safety concept.

5.4 ConSert derivation

At this point, all domain-level engineering activities have been finished, which enable different truck manufacturers to realize collaboration roles in isolation yielding a safe collaboration though. This section focuses on the creation of ConSerts, which assemble the safety-relevant information from the existing artifacts into a runtime representation so that an automated check can be performed at runtime, whether a safe collaboration can be guaranteed or not. While *Section 5.4.1* explains the relation between the domain-level artifacts, system-level artifacts and ConSerts conceptually, *Section 5.4.2* presents the ConSert models for the running example.

5.4.1 Transition from domain to systems engineering

On the domain engineering level, a repository of the following artifacts has been created for the platooning collaboration so far:

1. Role definitions
2. Role configurations describing different possible service interfaces of a role
3. Collaboration scenarios considering specific configuration combinations for each role
4. Safety property types attached to the services of a role configuration
5. Safety property refinements describing concrete deviation bounds for services within a collaboration scenario

A truck manufacturer with the goal to turn either a new or an existing truck of his fleet platooning-ready can make use of the domain repository in the following way. Firstly, he has to select the role(s) and the role configurations the truck under development should support. This decision is mainly dependent on existing hardware, cost and quality guarantees that are achievable with the selected configurations. For a specific configuration, the guaranteed quality attributes are determined through the set of provided and required services. However, variable safety guarantees and therefore variable functional performances might be given for a single

provided service dependent on the number of selected safety property refinements. The domain-level repository from which the manufacturer can choose is visualized on the left side in *Figure 5.15*.

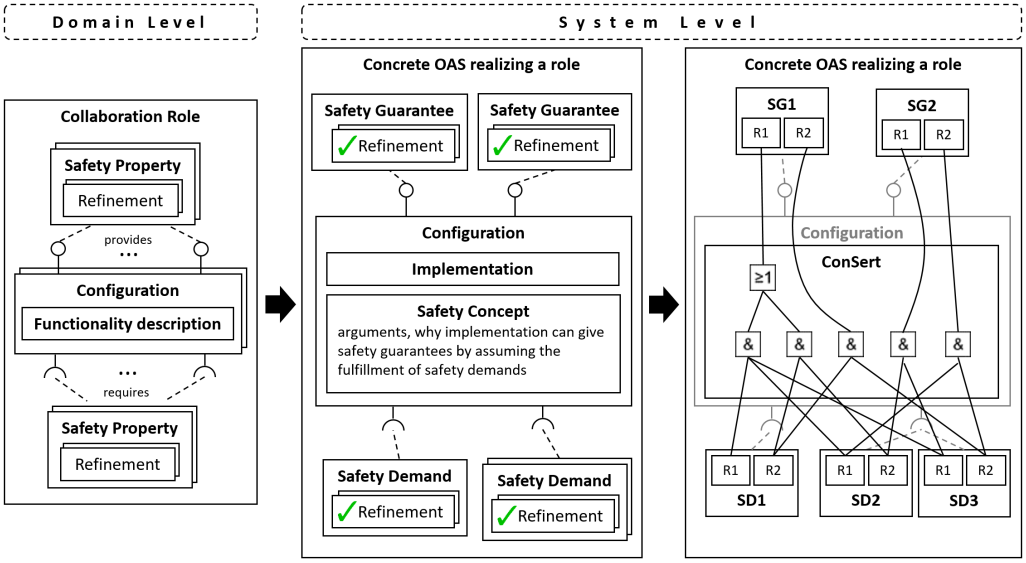


Figure 5.15: OAS development phases

Having performed the selection of *which* functionality has to be realized (roles, configurations) and *how well* the achievable performance at runtime should be (safety property refinements), the manufacturer has to technically design and implement the functionality. Thereby, the manufacturer can assume that there exist role configurations for other roles matching the service interface of his selected one(s). This includes that the concrete implementation can build upon the fulfillment of external safety demands, i.e. that for required services, maximum deviations bounds are guaranteed. Thus, the manufacturer has to provide a realization of the configuration's provided services that satisfy the safety guarantees associated with these services.

Note that the viewpoint that a manufacturer can assume that roles exist with matching service and safety interfaces is a fundamental difference to the original notion of ConSerts as defined in [5]. Refer to the discussion in *Section 6.2* for more details on the reasons, why we think that a top-down definition of ConSerts might be more valuable for some collaborations than the originally proposed bottom-up version from [5].

In parallel to the design of the technical solution, the manufacturer has to develop a safety concept providing a sound argumentation, how and why the concrete implementation is able to provide the safety guarantees with the required quality specified as part of the refinements. General considerations of how such a safety concept could look like have been given in

Section 5.3. In particular, the safety concept will be considerably different in nature and extent for roles providing application services in contrast to roles acting only as "data providers" with minimal safety responsibility. The argumentation of how the safety guarantees are met by the implementation, consists of a top-down graph structure that relates safety guarantees at its top to safety demands at the bottom, which are required to be fulfilled from other OAS. The detailed argumentation is thus the connecting piece in the middle of the graph.

As soon as implementation and safety concept/argumentation are finished, safety standards like ISO 26262 require a certification procedure, where external certification bodies evaluate, whether design, implementation and safety concept fulfill the standard's requirements. If this is the case, a safety certificate is issued for the examined item and finally it can be released to the market. In the context of closed systems, the examined item is certified with respect to functionality that is self-contained and completely available during the certification. However, the situation is different for OAS, because the certification of a certain configuration can only happen conditionally, i.e. the fulfillment of safety guarantees can only be certified under the condition that external safety demands of the configuration's services are fulfilled as well. This mapping between safety guarantees and safety demands of a configuration is represented by ConSerts (right side of *Figure 5.15*). Thus, a ConSert can be thought of either as an abstraction of a configuration's safety concept or as a part of the more general safety case that only includes the information, which is required to check at runtime, whether the safety interfaces of two different OAS are compatible with each other. For both OAS, the fulfillment of their respective safety interface has been pre-certified at development time and a ConSert has been issued. Hence, the detailed safety concepts do not have to be present at runtime.

The basic model elements of ConSerts are Boolean operations that establish the relation between safety guarantees and safety demands. In *Figure 5.15*, it can be observed that the quality resulting from the refinement R1 of safety guarantee SG1 can be guaranteed in two situations: Either all of the safety demands SD1 - R1, SD2 - R1 and SD3 - R1 are fulfilled by an external OAS or SD1 - R2 and SD2 - R2 are satisfied. Thus, in addition to the variability of choosing among different configurations, ConSerts offer an additional variability dimension that considers only the safety properties of services.

5.4.2 Platooning ConSerts definition

This section presents the ConSerts for the leader and follower configurations of platooning scenario S2, whose service architecture has been defined in *Section 4.5.2*. A graphical representation of both ConSerts is visualized in *Figure 5.16*.

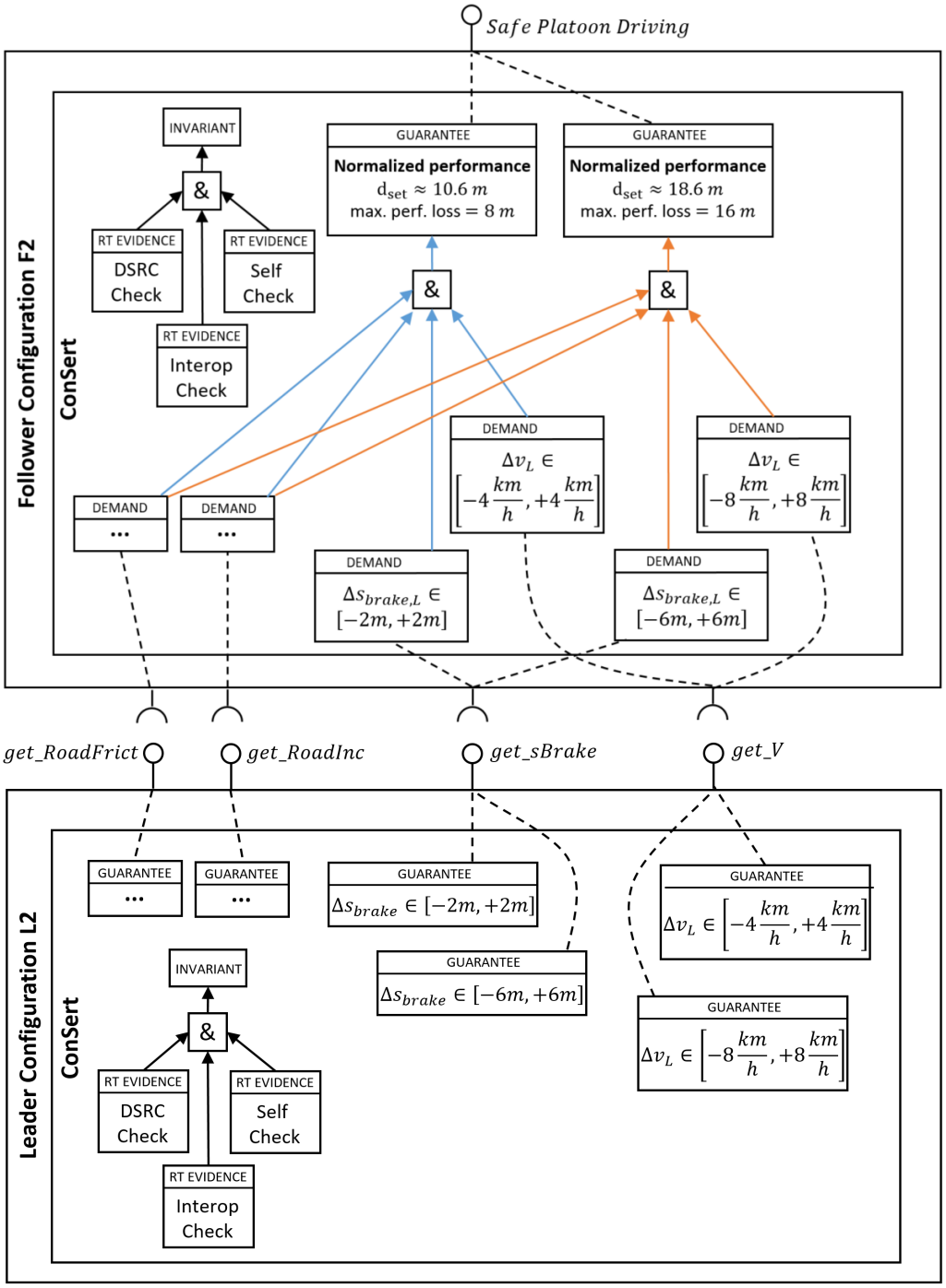


Figure 5.16: ConSerts for leader and follower configurations

Both ConSerts have in common that in order to deliver a guarantee at all, certain invariants have to be ensured. These include the availability and a proper setup of the DSRC communication stack, a check whether the collaboration partner "speaks" the same protocol for the automatic composition and evaluation of ConSerts and finally a self-check, whether the OAS itself can provide sufficient information about its realized roles, configurations and ConSerts. The invariants have been modeled as runtime evidences, because these are no classical demands to other OAS and can only be evaluated immediately before a collaboration shall be initiated. Unfortunately, the simulative safety property refinement could not be carried out for the services *get_RoadFrict* and *get_RoadInc*, because the given computation power for their simulation was not available (refer to *Section 5.2.4* for more details). Therefore, their associated safety demands and guarantees do not have concrete refinement data.

Considering the leader's ConSert, it becomes apparent that the leader does neither require any service nor any safety demand from other roles. This is the case, because the leader is the role with minimum context knowledge in this scenario and thus only acts as a "data provider". However, the leader provides several safety guarantees for each provided service with different value deviation ranges. We chose two different guarantees per service to increase chances that a collaboration can finally materialize on highway. In addition, it can be expected that the smaller the guaranteed value deviations are for the services, the better the functional performance of the collaboration will be, if an OAS with a compatible ConSert for the follower role is present at platoon building time.

This performance increase can be observed when looking at the ConSert of the follower configuration, because the follower role is responsible for the provision of the application service in the given scenario. The application service *Safe Platoon Driving* can be provided with different guaranteed qualities. While safety is guaranteed in both cases, the functional performance of the collaboration, i.e. the required minimum inter-truck distance d_{set} as well as the maximum performance loss, vary among the guarantees. The requirements for the fulfillment of each guarantee are modeled by two separate ConSert trees (colored blue and orange in *Figure 5.16*). While the blue ConSert tree yields the better performance, it also demands tighter deviation ranges for the leader's provided speed v_L and brake distance $s_{brake,L}$.

The normalized performance values for the minimum required inter-truck distance have been computed with equal truck parameters for both trucks as given in *Table 5.2*. The maximum performance loss represents the additionally required distance to compensate potential deviations. If the maximum performance loss is subtracted from d_{set}, the resulting value of 2.6 meters is the distance that would be safe, if no deviations occurred at all. The maximum performance loss occurs in the case, where the least safety-critical deviations are present at the required services of the follower configuration, i.e. where Δv_L =-4 or -8 km/h and $\Delta s_{brake,L}$ =-2 or -6 m.

Current speed	Road slope	Road friction coeff.
90 km/h	0 %	1
Truck mass (incl. cargo)	Max. deceleration	Follower reaction time
40t	8 m/s²	50 ms
Air resistance area	Roll res. coeff.	Rot. mass factor
4.68 m²	0.007	1.0
Brake response time	Brake dwell time	
0.15 s	0.3 s	

Table 5.2: Assumed truck parameters for determining the normalized performance

Regarding the goal of the *European Truck Platooning Platooning Challenge* [23] that inter-truck distances of around 9 meters should be reached during the nominal collaboration, it can be stated that the safe distances, which can be guaranteed for the application service *during the occurrence of deviations*, still enable significant fuel-saving.

Since safety guarantees and safety demands conceptually represent safety requirements, they should in theory have an assigned integrity level, e.g. an automotive safety integrity level (ASIL), indicating the required confidence with which the requirements have to be fulfilled. However, it is evident that in the given ConSerts all guarantees and demands have to be rated with the highest ASIL level D or in theory with 100% required confidence, since every violation of the deviation boundary intervals might yield a frontal crash causing severe harm.

6 Discussion and conclusion

6.1 Summary

In this thesis the Consert approach has been applied and evaluated for engineering safe collaborations of open adaptive systems. Concretely, the conducted case study exemplified the approach for truck platooning as an application scenario for the automotive domain. Since the derivation of ConSerts in general requires a reference system model before any analysis of its deviations can be carried out, this thesis was structured according to these two major activities: Firstly, the specification of a *safe nominal behavior* did explicitly exclude the consideration of failure occurrences and instead focused on the safety of the intended functionality. The essential output artifact of the safe nominal behavior specification is the service architecture describing how the collaboration's functionality is distributed among the collaborating OAS and how the resulting collaboration interface formalized in the shape of exchanged services is defined. Secondly, the effects of random hardware and systematic software failures on the collaboration have been analyzed and finally constrained as part of the *safe fail behavior* specification. These constraints define maximum boundaries on safety-critical service deviations, which are documented as quantified safety properties for each service of the collaboration interface. Finally, ConSerts have been derived by logically mapping safety guarantees of provided services of a role configuration to safety demands of its required services. The activities and created artifacts of the whole engineering process are visualized in *Figure 6.1*.

At the beginning of the safe nominal behavior derivation, we identified that the results of a preliminary hazard and risk analysis, i.e. potential collaboration accidents, are required as an essential input to perform a systematic functional decomposition yielding a safe nominal behavior by construction. Based on the identified accidents, the collaboration's state space has been partitioned into safe and unsafe spaces, where a safe space is defined physically by so-called safe conditions preventing an accident if these conditions are fulfilled. Afterwards, the functional decomposition was performed by using a horizontal refinement strategy breaking down the safe condition's physical entities both along and against the signal flow, until concrete measurable system inputs and controllable system outputs have been identified. Since the functional model does neither consider variability aspects of the single OAS nor the inherent functionality distribution within OAS collaborations, an abstraction layer has been introduced between functional model and concrete truck systems to account for these aspects. Concretely, this abstraction layer consists of collabora-

tion roles, role configurations (potential adaptation variants of roles) yielding different service interfaces and collaboration scenarios (concrete combinations of role configurations for all required roles), which are suitable for building a reusable domain-specific repository for automotive platooning in general. Considering concrete collaboration scenarios, the functions have been systematically deployed to collaboration roles based on a set of heuristics, which aim on the one hand at keeping the resulting service interface as slim as possible and on the other hand at producing single roles taking the majority of functional responsibilities. This approach also supported the assignment of the collaboration's safety responsibility to a single role in the shape of the application service. After having deployed the functions to roles and their configurations, the service architecture is completely defined through collaboration scenarios that contain role configurations with well-defined functional service interfaces and functional responsibilities.

Having defined a safe nominal behavior for platooning through the service architecture on the domain-level with an agreement between companies of that domain, the next step was to examine and constrain the impact of failures on the nominal collaboration. Therefore, a safety analysis of the collaboration interface's functional service types has been carried out. The results of the HAZOP-based analysis led to safety property types defining qualitative safety-critical deviations of each functional service type. After the safety-critical deviations have been analyzed qualitatively, they were quantified through a simulative approach. The quantification procedure is necessary to define for each safety property type, how much deviation can be tolerated for a service without risking an unsafe collaboration. The simulative approach required the creation of a simulation model for platooning in Matlab/Simulink, which was used for the simulation of deviation vectors for all services of the collaboration interface. The simulation results acted as the basis for the selection of safety property refinements by choosing those deviation vectors that yielded a safe collaboration during simulation. In this way, safety can be guaranteed for a collaboration, if the service value deviations stay within the specified bounds. In contrast to functional adaptation variants, which can be expressed through role configurations, we found that safety property refinements introduce another dimension of variability, where deviation tolerances can be traded off against the collaboration's functional performance. Having in place role configurations with a clearly defined interface of provided and required services including their refined safety properties, ConSerts are defined as an a part of each role's system-level safety case containing an argumentation how safety guarantees are fulfilled by the technical system implementing the role under the condition that potential safety demands are fulfilled by other roles. However the technical implementation of role configurations as well as the creation of the concrete system-level safety concept/safety case have been out of the thesis scope and are therefore greyed out in *Figure 6.1*.

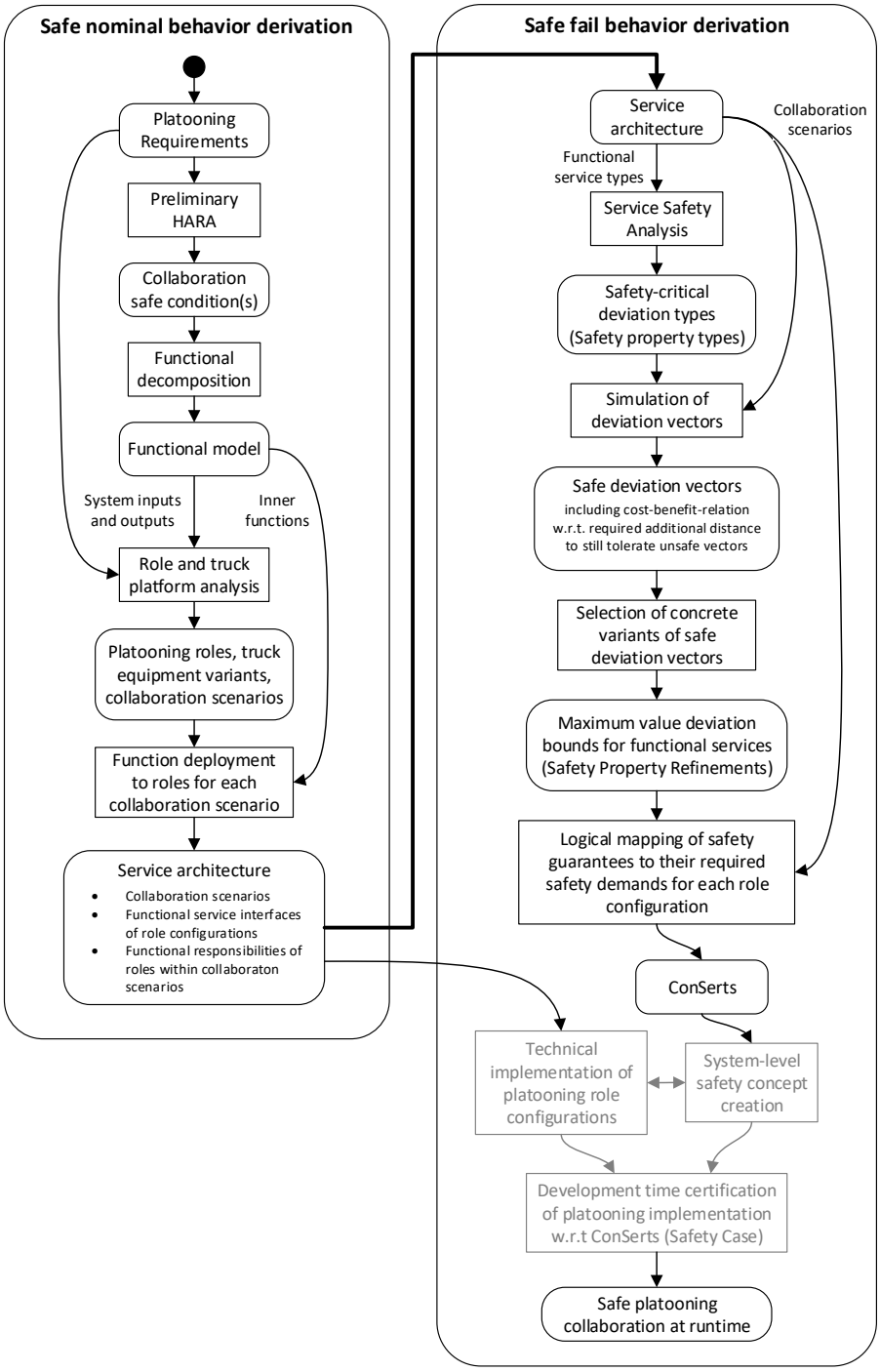


Figure 6.1: Summary of engineering process for safe OAS collaborations

6.2 Discussion

This section critically discusses, to which degree the thesis goals as set out in *Section 1.3* have been fulfilled. In addition, a selection of interesting findings and lessons learned are stated and discussed.

Goal 1 – Systematic engineering of safe nominal behavior for the collaboration between two trucks

The hazard and risk analysis approach as recommended by ISO 26262 is insufficient for OAS collaborations.

The systematic execution of the preliminary HARA according to ISO 26262 yielded problems, because it is hardly possible to determine, whether a truck behavior like an acceleration is malfunctioning and thus safety-critical for the collaboration without the consideration of the other truck's behavior, too. In this way, the standard's recommendation to restrict the consideration scope of the safety analysis to a single vehicle or its subsystem does not yield valuable results. In addition, failure modes arising from the collaboration itself, e.g. from the communication infrastructure, are thus missed, because only the trucks would be considered proper items in the sense of the standard. Within the case study, the solution approach for this problem consisted of a change of the safety analysis starting point: Instead of examining potential safety-critical effects of single truck behaviors in certain situations on the collaboration, a shift towards a deductive viewpoint has been carried out, where the analysis is started at the safety-critical states of the collaboration and afterwards traced back to each single truck being considered.

The functional decomposition is dependent on a successful formal characterization of the collaboration's safe state space.

For the platooning collaboration, the partitioning of the collaboration's state space into safe and unsafe spaces was relatively easy by considering the inter-truck distance before an emergency braking maneuver of the platoon as a means for formalizing the state space. Since the distance is a one-dimensional quantity and the safe space is continuous, a single mathematical condition is sufficient for a complete safety-specific characterization of the collaboration's state space, i.e. if the condition is met, the collaboration is safe, and vice versa. However, it can be expected that a safety-specific characterization of the collaboration's state space can become complex for other collaborations. This can happen in particular, if either multiple potentially non-scalar quantities are needed for a sufficient formalization of the safe space or if the safe space itself is distributed into multiple smaller spaces that are not adjoining each other. Apart from the fact that this might require deep mathematical and physical knowledge, the amount of required conditions to distinguish a safe from an unsafe collaboration state could reach a significant number, if they can be derived at all. In this regard, basing the functional decomposition on safe conditions

is assessed as a slight drawback of the approach, since it is not known yet, if the derivation of safe conditions is possible for other collaboration scenarios.

Detailed and realistic functional models are required for demonstrating the effectiveness of engineering methods and for a proper interpretation of prediction results of quality characteristics.

Concretely, the functional decomposition for platooning demanded a decision on how realistic the scenario should be modeled with respect to physics. The safe condition contains a relation of required stop distances of both trucks for an emergency braking maneuver to compute a safe distance. Due to the fact that the stop distance is dependent on the underlying physical model, the potentially usable models range from very simple to more complex and hence more realistic. Early benchmark computations with simple truck motion models yielded minimum safe distances between the trucks larger than 30 meters. These results suggested that the model was too simple for guaranteeing safe truck platooning with distances around 9 meters during nominal cruising with speeds of 90 km/h, which has been set out as a goal for the European Truck Platooning Challenge [23], into which this thesis was embedded. Consequently, a more sophisticated non-linear dynamic truck motion model has been developed that finally yielded a minimum safe distance of 2.6 meters during nominal cruising at speeds of 90 km/h. This result was accepted as realistic enough, since a game-theoretical approach in [39] yielded comparable results of 1.2 - 2 meters. For the ability to finally demonstrate whether the toleration of service failures as part of the ConSert approach yielded acceptable performance losses still enabling fuel-efficient platooning, the realistic functional model was mandatory.

The assignment of the application service does not fit into the service architecture derivation activity.

During the derivation of the service architecture, the reasonable assignment of the application service provision to a specific role proved to be a difficult task, because the application service is a construct that is in the first place considered for assigning the responsibility of guaranteeing the collaboration's safety to a single role. However, the assignment of the app. service has an effect only then, when the system-level safety concept has to be developed for the role taking the safety responsibility. Other than that, the application service is considered as a root entity for the technical composition and evaluation of ConSerts at runtime. Although the construction of a safe nominal behavior is indeed based on safety considerations (=safe conditions), it is questionable, if the application service provision has to be assigned already to a role during the service architecture derivation for the mentioned reasons. In order to still provide an app. service assignment in this thesis, the heuristics for the function deployment to roles were designed in a way so that as much functionality as possible is deployed to a single collaboration role. This consequently yields

a single role with maximum context knowledge, which suggests the assignment of the safety responsibility and therefore the application service to that role, too. In this way, this role will have to realize a kind of "application service controller" whose inputs are either services provided by the own OAS platform (sensors or actuators) or services provided by other roles. The output of such a controller would be a clear decision whether a safe collaboration can be guaranteed. If this is not the case, it has a second output that is able to trigger and control a transition to a safe state.

Goal 2 – Methodological derivation and construction of related ConSerts models for guaranteeing a safe collaboration during failure

No reasonable quantification strategy has been found for the refinement of omission safety property types.

Although service deviations due to a permanent omission of the service provision have been identified as being potentially safety-critical for the platooning collaboration, no reasonable strategy for their safety property quantification has been found. We think that permanent omission failures cannot be safely tolerated without the use of redundancy concepts. However, redundancy concepts have not been explicitly studied in the course of this thesis and thus, the reasonable quantification of omission safety properties stays an open issue.

Numerical simulation as a means for the safety property quantification is not scalable.

A major drawback of the numerical simulation approach is that each service deviation vector, i.e. a set of concrete values for the deviations of all services of the collaboration interface, has to be simulated in a separate simulation run. This leads to serious performance problems due to the required size and granularity of deviation ranges that need to simulated to perform a reasonable safety property quantification. However, recently published research in the context of uncertainty handling for input signals of mixed-signal systems proposed a much more efficient simulation approach that is based on a symbolic rather than numerical simulation [40]. The approach enables an efficient simulation of a *value range* for each signal in one simulation run instead of using concrete values. With respect to the applicability for the safety property quantification, this means that one simulation run could suffice for the simulation of the desired value deviation ranges for all services suggesting a significant performance improvement for the quantification procedure.

ConSerts enable a dynamic optimization of runtime qualities.

The possibility to provide varying safety guarantees for a single service within a ConSert enables a dynamic optimization the overall collaboration's runtime qualities. This claim is supported by the provided ConSerts for truck platooning (*Section 5.4.2*), because the application service can

be provided safely with two guarantees optimizing a different quality each. While guaranteeing a higher functional performance leads to tighter boundaries for tolerated service deviations, higher tolerated service deviations in turn cause a degradation of the achievable functional performance. With respect to truck platooning, the functional performance can be expressed through the minimum required inter-truck distance, which has a direct relation to the saved fuel of the follower truck, since the lower distance can be maintained safely, the higher the saved fuel will be. In a similar way, the tolerance of broader or tighter deviation ranges can be potentially related with required development cost of the systems that have to provide services with more or less tighter deviation ranges.

The ConSerts for role configurations of concrete collaboration scenarios should be derived as pre-engineered compatible safety interfaces that enable a safe collaboration in a top-down manner.

During the execution of the case study, it was found that ConSerts as proposed in [5] are capable of serving as more than mere runtime representations of conditional safety certificates being issued for finished technical OAS implementations at development time. In addition, they could be as well used as an abstraction defining the safety interface of a modular system-level safety concept that the manufacturer of an OAS has to adhere to during the technical implementation. This additional use case however contradicts the notion of [5], where ConSerts are issued for OAS implementations in a bottom-up manner to *prohibit unsafe collaborations* at runtime in the first place. In contrast, the usage of ConSerts as mandatory safety interfaces in a top-down manner could be an approach that rather *enables safe collaborations* directly from the start of the collaboration engineering on the domain-level. Since the agreement among a majority of an application domain's companies upon pre-engineered collaboration scenarios is required anyway for bringing the idea of a "safety domain repository" to success, the safety interface definitions in the shape of ConSerts could be thus an essential entity of this repository as well.

6.3 Conclusion

This thesis contributed to the body of scientific knowledge in that the ConSert approach as a novel means for the runtime safety assurance of open adaptive systems has been applied and evaluated for a truck platooning scenario in the automotive domain. The operationalization of ConSerts consists of an analysis and restriction of failure-caused system behavior deviations. It was found that before the operationalization can be carried out, a nominal and safe description of the intended collaboration behavior is required, which can act as a reference for the definition of behavioral deviations. Thus, *Chapter 4* provided a detailed guideline for the required activities to engineer a safe nominal behavior description for truck platooning being suitable for the subsequent application of the ConSert ap-

proach. Based on the safe nominal behavior specification in the shape of a service-oriented architecture, the ConSert approach was finally applied in *Chapter 5*.

Regarding limitations of the proposed engineering approach other than the ones mentioned in the discussion *Section 6.2*, it can be stated that the considered platooning scenario only consisted of two collaboration roles having a relatively low amount of exchanged services. Hence, it is not clear, whether the guideline application scales properly for collaborations with more than two roles that have extensive collaboration interfaces. With respect to a general applicability of the ConSert approach, the lack of proper quantification strategies for other safety property types than value deviations is deemed the biggest persistent methodological gap.

In conclusion, this thesis provided on the one hand evidence that the ConSert approach as such can be successfully applied for truck platooning collaborations. On the other hand, the engineering guidelines that have been proposed for the specification of safe nominal and safe fail behaviors comprise a wider scope so that we think they can be easily adapted for similar automotive collaboration scenarios.

6.4 Future research recommendations

Based on the potential usage of ConSerts as top-down safety interfaces on the domain level as proposed in *Section 6.2*, a possible area for future research could be a further investigation of how concrete technical realizations and system-level safety concepts can be developed for the truck platooning collaboration. In addition, this approach would give evidence about the suitability of the derived domain-level artifacts in this thesis for the concrete realization of OAS.

Further, as motivated in *Section 6.2*, the evaluation of the symbolic simulation approach proposed in [40] is recommended for the future to improve the simulation performance of the safety property quantification procedure.

Lastly, an area of future work that is in particular interesting for practitioners could be the integration of the ConSert approach into common tool chains for model-based safety assurance. This includes a dedicated support for modeling both safe nominal and safe fail behavior specifications, where an integration with simulation tools like Matlab/Simulink is considered beneficial.

References

[1] H. Proff, J. Schönharting, D. Schramm, and J. Ziegler, eds., *Zukünftige Entwicklungen in der Mobilität*. Springer Science + Business Media, 2012.

[2] McKinsey, "Delivering Change - Study on the future of commercial vehicles," 2016. `https://www.mckinsey.de/deliveringchange` – Accessed: 16.09.2016.

[3] S. Kemmann, *SAHARA - A Structured Approach for Hazard Analysis and Risk Assessments*. PhD thesis, University of Kaiserslautern, 2015.

[4] B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A New Component Concept for Fault Trees," in *Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software - Volume 33*, SCS '03, (Darlinghurst, Australia, Australia), pp. 37–46, Australian Computer Society, Inc., 2003.

[5] D. Schneider, *Conditional Safety Certification for Open Adaptive Systems*. PhD thesis, University of Kaiserslautern, 2014.

[6] M. Jamshidi, "System of systems engineering - New challenges for the 21st century," *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, pp. 4–19, may 2008.

[7] M. Jamshidi, *Systems of Systems Engineering: Principles and Applications*. CRC Press, 2008.

[8] M. DiMario, J. Boardman, and B. Sauser, "System of systems collaborative formation," *IEEE Systems Journal*, vol. 3, pp. 360–368, sep 2009.

[9] "ETSI EN 302 665 – Intelligent Transport Systems (ITS); Communications Architecture," Standard, European Telecommunications Standards Institute, 2010.

[10] "ETSI TS 102 637 – Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications - Part 1 to 4," Standard, European Telecommunications Standards Institute, 2010.

[11] "SAE J 2945/1 – On-Board System Requirements for V2V Safety Communications," Standard, Society of Automotive Engineers, 2016.

[12] "SAE J 2735 – Dedicated Short Range Communications (DSRC) Message Set Dictionary," Standard, Society of Automotive Engineers, 2016.

[13] H. Foster, A. Mukhija, D. S. Rosenblum, and S. Uchitel, "A Model-Driven Approach to Dynamic and Adaptive Service Brokering Using Modes," in *Lecture Notes in Computer Science*, pp. 558–564, Springer Science + Business Media.

[14] M. Röckl, J. Gacnik, and J. Schomerus, "Integration of Car-2-Car Communication as a Virtual Sensor in Automotive Sensor Fusion for Advanced Driver Assistance Systems," in *Proceedings. Springer Automotive Media. FISITA 2008*, 2008.

[15] M. Wagner, D. Zobel, and A. Meroth, "SODA: Service-Oriented Architecture for Runtime Adaptive Driver Assistance Systems," in *2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing*, Institute of Electrical and Electronics Engineers (IEEE), jun 2014.

[16] Object Management Group (OMG), "Service Oriented Architecture Modeling Language (SoaML)," 2012.
`http://www.omg.org/spec/SoaML/` – Accessed: 29.10.2016.

[17] D. Schneider and M. Trapp, "Engineering Conditional Safety Certificates for Open Adaptive Systems," *IFAC Proceedings Volumes*, vol. 46, no. 22, pp. 139–144, 2013.

[18] C. Bergenham, H. Pettersson, E. Coelingh, C. Englund, S. Shladover, and S. Tsugawa, "Overview of platooning systems," in *Proceedings of 19th ITS World Congress, Oct 22-26, Vienna, Austria*, 2012.

[19] J. Axelsson, "Safety in Vehicle Platooning: A Systematic Literature Review," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2016.

[20] J. Nilsson, C. Bergenhem, J. Jacobson, R. Johansson, and J. Vinter, "Functional Safety for Cooperative Systems," in *SAE Technical Paper Series*, SAE International, apr 2013.

[21] C. T. Featherstone and M. V. Lowson, "Safety of automated passenger vehicle platooning," in *Proceedings of the 12th World Congress on Intelligent Transport Systems*, 2005.

[22] C. Heinzemann, D. Schubert, S. Dziwok, U. Pohlmann, C. Priesterjahn, C. Brenner, and W. Schäfer, "Railcab convoys: An exemplar for using self-adaptation in cyber-physical systems," tech. rep., Software Engineering Group, Heinz Nixdorf Institute, University of Paderborn, 2015.

[23] Dutch Government, "European Truck Platooning Challenge," 2016.
`https://www.eutruckplatooning.com/` – Accessed: 14.09.2016.

[24] Springer Professional, "Europäische Kolonnenfahrt zeigt Potenziale der Lkw-Vernetzung," 2016.
`https://www.springerprofessional.de/nutzfahrzeuge/automatisiertes-fahren/europaeische-kolonnenfahrt-zeigt-potenziale-der-lkw-vernetzung/9975176` – Accessed: 16.09.2016.

[25] Straßenverkehrsordnung Deutschland (StVO), "Mindestabstand auf Autobahnen nach § 4 Abs. 3."

[26] Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, "Automotive Intelligent User Interfaces (IUI)."
`http://automotive.dfki.de/content/index.php/de/technology/driver-assistance-systems-adas` – Accessed: 29.10.2016.

[27] "ISO 26262:2011 – Road vehicles – Functional safety," Standard, International Organization for Standardization, Geneva, Switzerland, 2011.

[28] R. Adler, P. Feth, and D. Schneider, "Safety Engineering for Autonomous Vehicles," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, Institute of Electrical and Electronics Engineers (IEEE), 2016.

[29] H.-L. Ross, *Functional Safety for Road Vehicles*. Springer Nature, 2016.

[30] R. J. R. Back and K. Sere, "Superposition refinement of reactive systems," *Formal Aspects of Computing*, vol. 8, pp. 324–346, may 1996.

[31] E. Hoepke and S. Breuer, eds., *Nutzfahrzeugtechnik*. Springer Nature, 2016.

[32] H. Winner, S. Hakuli, F. Lotz, and C. Singer, eds., *Handbuch Fahrerassistenzsysteme*. Springer Science + Business Media, 3 ed., 2015.

[33] M. McIntyre, T. Ghotikar, A. Vahidi, X. Song, and D. Dawson, "A Two-Stage Lyapunov-Based Estimator for Estimation of Vehicle Mass and Road Grade," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 3177–3185, sep 2009.

[34] Continental, "New sensor fusion approach recognizes rain, snow and ice on the road," 2010.
`http://www.continental-corporation.com/www/pressportal_com_en/themes/press_releases/3_automotive_group/interior/press_releases/pr_2010_10_12_sensorfusion_en.html` – Accessed: 18.11.2016.

[35] P. Feth and R. Adler, "Service-based Modeling of Cyber-Physical Automotive Systems: A Classification of Services," in *Workshop CARS 2016 – Critical Automotive applications: Robustness & Safety* (M. Roy, ed.), 2016.

[36] F. Möhrle, M. Zeller, K. Höfig, M. Rothfelder, and P. Liggesmeyer, "Automating Compositional Safety Analysis Using a Failure Type Taxonomy for Component Fault Trees," in *Risk, Reliability and Safety: Innovating Theory and Practice* (Walls, Revie & Bedford, ed.), Taylor and Francis Group, 2017. Pre-print version.

[37] MathWorks, "Matlab/Simulink - Simulation Perfomance Modes," 2016.
`https://de.mathworks.com/help/simulink/ug/how-the-acceleration-modes-work.html` – Accessed: 03.12.2016.

[38] C. Gold, D. Dambock, L. Lorenz, and K. Bengler, ""Take over!" How long does it take to get the driver back into the loop?," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, pp. 1938–1942, sep 2013.

[39] A. Alam, A. Gattami, K. H. Johansson, and C. J. Tomlin, "Guaranteeing safety for heavy duty vehicle platooning: Safe set computations and experimental evaluations," *Control Engineering Practice*, vol. 24, pp. 33–41, mar 2014.

[40] C. Radojicic, *Symbolic Simulation of Mixed-Signal Systems with Extended Affine Arithmetic*. PhD thesis, University of Kaiserslautern, 2016.

A Appendix

A.1 Differential equations mathematical solution

This section shall give some more insight on the mathematical derivation of explicit solutions for the differential equations for the different braking phases of an emergency braking maneuver as defined in *Section 5.2.2*. The now generalized 1st order, 2nd degree inhomogeneous differential equations (DE) with constant coefficients are given in equations **(A.1)** – **(A.3)**.

$$v'(t) = \begin{cases} K_1 \cdot v^2(t) + K_2 & 0 \leq t \leq t_1 \quad \textbf{(A.1)} \\ K_1 \cdot v^2(t) + K_3 \cdot t + K_2 & t_1 < t < t_2 \quad \textbf{(A.2)} \\ K_1 \cdot v^2(t) + K_4 & t_2 \leq t \leq t_3 \quad \textbf{(A.3)} \end{cases}$$

with $K_i =$ contants

While equations **(A.1)** and **(A.3)** have exactly the same shape apart from different constants, **(A.2)** contains an additional term linear in t. In general it can be said that the solution procedure as well as the explicit solution of DEs of the shape $v'(t) = K_1 \cdot v^2(t) + K_2 \cdot t + K_3$ is much more complex than those of DEs of the shape $v'(t) = K_1 \cdot v^2(t) + K_2$, which will be shown in the following. Both solutions have been validated for their correctness with the mathematical toolboxes of *Matlab* and *Wolfram's Mathematica* to make sure the platooning system's realization in *Matlab/Simulink* is correct from a mathematical point of view.

Solution procedure for DE $\mathbf{v'(t) = K_1 \cdot v^2(t) + K_2}$

$v' = K_1 \cdot v^2 + K_2$

Separation of variables yields:

$$\mathrm{d}v = \left(K_1 \cdot v^2 + K_2\right) \mathrm{d}t$$

$$\Leftrightarrow \mathrm{d}v = K_2 \cdot \left(1 + \frac{K_1}{K_2} \cdot v^2\right) \mathrm{d}t$$

$$\Leftrightarrow \frac{\sqrt{\frac{K_1}{K_2}}\mathrm{d}v}{1 + \left(\sqrt{\frac{K_1}{K_2}} \cdot v\right)^2} = \pm\sqrt{K_1 \cdot K_2}\mathrm{d}t$$

Substitution of $\sqrt{\frac{K_1}{K_2}} \rightarrow \omega$:

$$\frac{\mathrm{d}\omega}{1+\omega^2} = \pm\sqrt{K_1 \cdot K_2}\mathrm{d}t$$

Given $\frac{\mathrm{d}\omega}{1+\omega^2} = \arctan'(\omega)$, the integration of both sides yields:

$$\arctan(\omega) + C_1 = \pm\sqrt{K_1 \cdot K_2} \cdot t + C_2$$

$$\Leftrightarrow \omega = \tan\left(\pm\sqrt{K_1 \cdot K_2} \cdot t + C_2 - C_1\right)$$

$$\Leftrightarrow \omega = \tan\left(\pm\sqrt{K_1 \cdot K_2} \cdot t + C_3\right) \text{ with } C_3 = C_2 - C_1$$

Back-Substitution of ω with the negative solution (we are braking!):

$$v(t) = \sqrt{\frac{K_2}{K_1}} \cdot \tan\left(-\sqrt{K_1 K_2} \cdot t + C_3\right)$$

C_3 can be determined with the intial condition $v(0) = v_0$
and yields the explicit solution for $v(t)$:

$$v(t) = -\sqrt{\frac{K_2}{K_1}} \cdot \tan\left(-\sqrt{K_1 K_2} \cdot t - \arctan\left(\frac{K_1}{K_2} \cdot v_0\right)\right)$$

Integration over time with the intial condition $s(0) = 0$
yields the explicit solution for $s(t)$ with only known constants:

$$s(t) = -\frac{\ln\left(|\cos\left(\sqrt{K_1 K_2 \cdot t}\right) + \frac{K_1}{K_2} \cdot v_0 \cdot \sin\left(\sqrt{K_1 K_2 \cdot t}\right)|\right)}{K_1}$$

Solution procedure for DE $\mathbf{v'(t) = K_1 \cdot v^2(t) + K_2 \cdot t + K_3}$

The solution procedure for the given DE is not carried out analytically here, since the explicit solution alone is filling approximately a single page alone. However, some information on the solution steps will be nevertheless given in the following.

The given DE is a specific form of *Riccati's differential equation*:

$$v'(t) = q(t)v^2(t) + p(t)v(t) + f(t) \qquad \text{Riccati DE}$$

with $q(t) = K_1$, $p(t) = 0$, $f(t) = K_2 \cdot t + K_3$

The application of any of both *Riccati transformation theorems*

$$u(t) = e^{\int q(t)v(t)dt} = e^{K_1 \int v(t)dt} \qquad \text{1st Riccati theorem}$$

$$v(t) = -\frac{u'(t)}{u(t)q(t)} = -\frac{u'(t)}{u(t)K_1} \qquad \text{2nd Riccati theorem}$$

yields the 2nd-order general homogeneous DE:

$$u'' - \left(p + \frac{q'}{q}\right)u' - fqu = 0$$

which is for the concrete scenario constants defined as:

$$u'' - (K_1 K_2 \cdot t + K_1 K_3) \cdot u = 0 \qquad \text{(A.4)}$$

Two linearly independent explicit solutions of the DE $u'' - x \cdot u = 0$ are the *Airy* functions of first and second kind.

$$Ai(x) = \frac{1}{\pi} \int_0^\infty \cos\left(\frac{t^3}{3} + xt\right) dt \qquad \text{Airy 1st kind}$$

$$Bi(x) = \frac{1}{\pi} \int_0^\infty \left(\exp\left(-\frac{t^3}{3} + xt\right) + \sin\left(\frac{t^3}{3} + xt\right)\right) dt \qquad \text{Airy 2nd kind}$$

Due to the specific form of $x = K_1K_2 \cdot t + K_1K_3$ in **(A.4)** and in particular its dependency on t, the explicit solution for equation **(A.4)** is a linear combination of the airy functions $Ai(x)$ and $Bi(x)$ as well as their derivatives $Ai'(x)$ and $Bi'(x)$. The now existing but very complex explicit solution of $u(t)$ can be substituted back with the **(1st Riccati theorem)**, which contains $\int v(t) = s(t)$, where $s(t)$ is the final solution for the stop distance that we wanted to compute.

Note that Matlab contains implementations of the Airy functions and their derivatives in its mathematical toolbox, so their usage can be realized in Simulink with a "Matlab Function Block".

A.2 Supplementary material

This section provides some additional material for the chapters of this thesis in an unordered way. At the respective locations of the thesis, the materials are referenced.

Safety Property Type	Deviation from correct service	Refinement parameters	Potential causes	Possible consequences for collaboration
Commission, too early and too late are not meaningful for services delivering continuous signals				
Functional service type get_RoadFrict (Road friction provision)				
Omission	No road friction value is received at the service consumer for more than y time units	Omission time y ms	Truck out of comm. range, comm. medium overload	Safety-critical, because Follower Stop Distance Determination lacks the required knowledge for computing its exact stop distance, thus an unsafe distance can arise. A switch to a worst-case value (e.g. $\mu = 1.2$) or immediate platoon dissolving are possible.
Too low	Received road friction is more than x % lower than the actual value	Maximum deviation x %	Leader sensor failure, Environmental noise	Not safety-critical: A lower perceived road friction value yields a longer follower stop distance and thus the safe distance set point is increased according to the platooning safe distance definition → distance increases and is thus safe
Too high	Received road friction is more than x % higher than the actual value	Maximum deviation x %		Safety-critical: A higher perceived road friction value yields a shorter follower stop distance thus the safe distance set point is decreased according to the platooning safe distance definition → distance decreases and might be unsafe
Functional service type get_RoadInc (Road inclination provision)				
Omission	No road inclination value is received at the service consumer for more than y time units	Omission time y ms	Truck out of comm. range, comm. medium overload, Inclination sensor failure	Safety-critical, because Follower Stop Distance Determination lacks the required knowledge for computing its exact stop distance, thus an unsafe distance can arise. A switch to a worst-case value (e.g. $q = 8\%$) or immediate platoon dissolving are possible.
Too low	Received road inclination is more than x % lower than the actual value	Maximum deviation x %	Inclination sensor failure (drift, bias, etc.), Environmental noise	Not safety-critical: A lower perceived road inclination value yields a longer follower stop distance and thus the safe distance set point is increased according to the platooning safe distance definition → distance increases and is thus safe
Too high	Received speed is more than x % higher than the actual value	Maximum deviation x %		Safety-critical: A higher perceived road inclination value yields a shorter follower stop distance thus the safe distance set point is decreased according to the platooning safe distance definition → distance decreases and might be unsafe

Figure A.1: Safety property types of service types *get_RoadFrict* and *get_RoadInc*

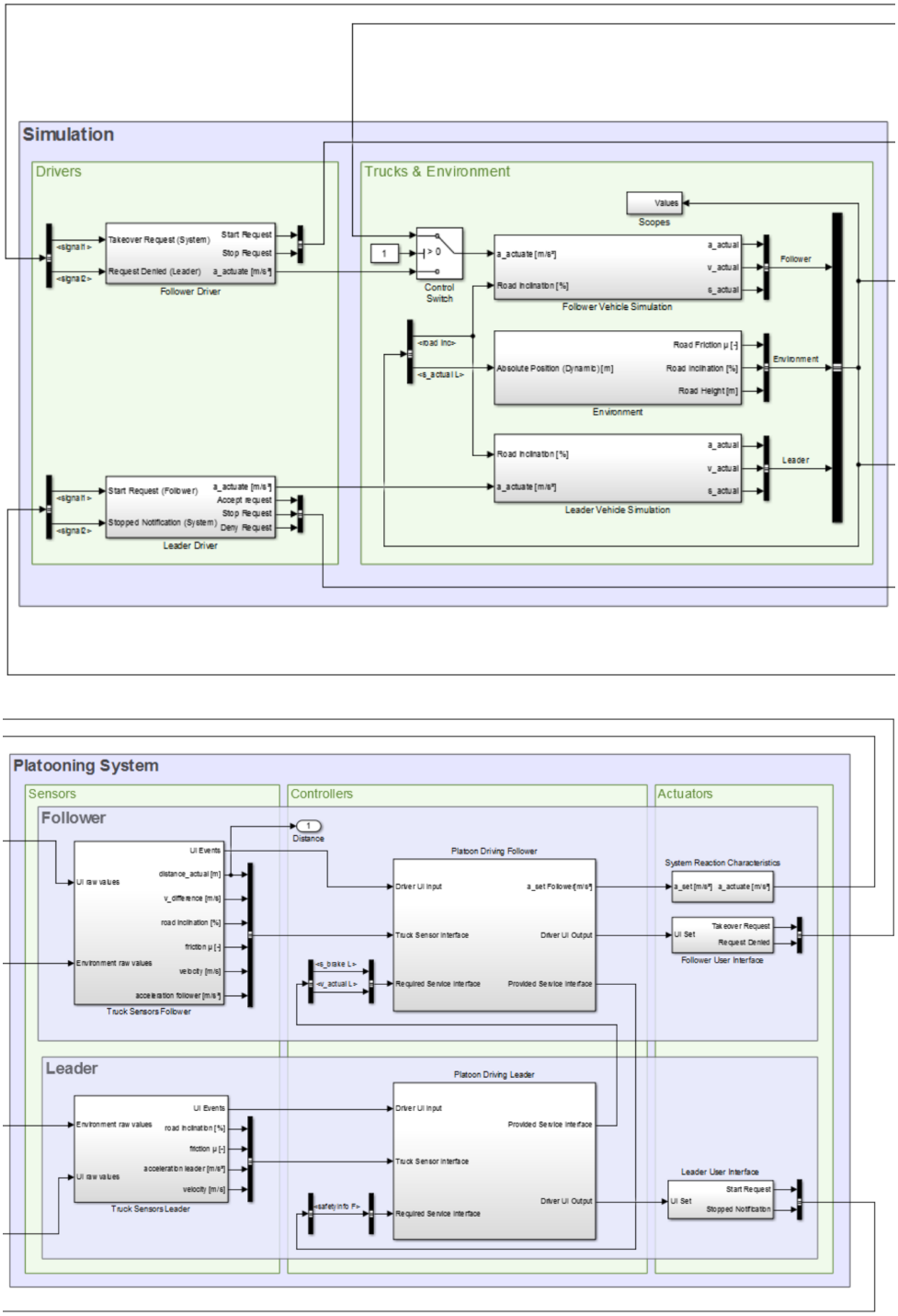


Figure A.2: Simulink model of drivers, trucks, environment (top) and platooning system (bottom)