Engineering of Runtime Safety Monitors for Cyber-Physical Systems with Digital Dependability Identities

Jan Reich[1(✉)], Daniel Schneider[1], Ioannis Sorokos[1], Yiannis Papadopoulos[2], Tim Kelly[3], Ran Wei[3], Eric Armengaud[4], and Cem Kaypmaz[5]

[1] Fraunhofer IESE, Kaiserslautern, Germany
{jan.reich,daniel.schneider,ioannis.sorokos}@iese.fraunhofer.de
[2] University of Hull, Kingston upon Hull, UK
y.i.papadopoulos@hull.ac.uk
[3] University of York, York, UK
{tim.kelly,ran.wei}@york.ac.uk
[4] AVL List GmbH, Graz, Austria
eric.armengaud@avl.com
[5] AVL Turkey, Istanbul, Turkey
cem.kaypmaz@avl.com

Abstract. Cyber-Physical Systems (CPS) harbor the enormous potential for societal improvement in terms of safety, comfort and economic efficiency. However, these benefits will only be unlocked if the safety of these systems can be assured with a sufficient level of confidence. Traditional safety engineering and assurance approaches alone cannot address the CPS-inherent uncertainties and unknowns induced by openness and adaptivity. Runtime safety assurance approaches such as *Conditional Safety Certificates* (ConSerts) represent novel means to cope with CPS assurance challenges by introducing modular and formalized safety arguments with variant support, thereby shifting the final safety certification step to runtime. However, the systematic engineering of ConSerts at design-time is a complex task which, up to now, has not been sufficiently addressed. Without systematic safety assurance at *both* design-time and runtime, CPS will hardly be assurable with acceptable confidence given the uncertainties and unknowns. In this paper, we present an engineering method for synthesizing ConSerts based on *Digital Dependability Identities* (DDI). The approach is demonstrated for a cooperative vehicle platooning function (CACC) from an industrial case study.

Keywords: Dynamic risk management · Runtime certification · Runtime safety monitor · Model-based safety engineering

1 Introduction

CPS Safety. Cooperative Cyber-Physical Systems (CPS) harbor enormous potential for societal improvement in terms of safety, comfort and economic efficiency. However, CPS

A. Casimiro et al. (Eds.): SAFECOMP 2020, LNCS 12234, pp. 3–17, 2020.
https://doi.org/10.1007/978-3-030-54549-9_1

functions will only be accepted by society if their safety is confidently assured. Justified belief that systems are free from posing an unacceptable risk to their environment must be created. Therefore, we need a safety argument, conveying a convincing story about why evidence, in the form of safety analyses, design measures and verification/validation results, supports the safety claim.

CPS Safety-related Uncertainties and Unknowns. Traditional safety assurance approaches, e.g. the automotive functional safety standard ISO 26262, assume that the complete set of evidence for supporting the safety claim can be generated at design-time. The CPS-inherent characteristics of openness and adaptivity pose a significant problem for traditional approaches because the amount of safety-relevant CPS context changes can hardly be anticipated completely at development time. These changes include CPS capability changes, triggered by e.g. sensor failure or changing cooperation partner capabilities, and environmental changes, such as vision range restriction or road friction changes due to weather. Thus, CPS complexity renders safety certification at development time with acceptable performance much harder.

Safety@Runtime. Runtime safety assurance approaches such as Conditional Safety Certificates (*ConSerts*) [1] represent novel means to cope with the CPS assurance challenges by shifting parts of the safety assurance process to the runtime, when all relevant uncertainties and unknowns can be resolved. Specifically, ConSerts allow the definition of modular safety concepts describing CPS cooperation variants. By making the guaranteed and demanded safety interface to other CPS systems and the environment explicit, these modular safety concepts are certifiable at design-time. At runtime, CPS constituents resolve those open ends by checking the compatibility of the safety interface and by monitoring runtime evidence required for safe operation.

Problem. ConSerts rely on solid design-time safety engineering and only shift the minimum necessary safety activities to runtime, i.e. safety interface matching and runtime evidence monitoring. Only limited research has yet examined this design-time engineering backbone; requirements for a concrete ConSerts-based engineering method remain an open research topic. Key aspects of such a method are a) a comprehensive assurance argument combining development time safety assurance with runtime safety assurance and b) a systematic design of ConSerts out of established development time safety artifacts.

Solution. *Digital Dependability Identities* (DDI) are an overarching solution framework for engineering dependable CPS, developed in the H2020 DEIS project. In this paper, we demonstrate the application of the DDI framework for an industrial case study of a cooperative platooning function, specifically focusing on the transition from design-time safety models (*Design-time DDIs*) to runtime safety models (*ConSerts*, which serve as *Runtime DDIs*). To that end, in Sect. 2, we introduce the overall idea of the DDI engineering framework for design-time and runtime DDIs. In Sect. 3, the framework is applied and discussed for a CPS-based platooning function. Section 4 discusses related work and Sect. 5 concludes the paper's scientific contribution.

2 Runtime DDI Engineering Approach Overview

This section introduces the DDI engineering approach for seamless dependability engineering of CPS functions by creating design-time DDIs and transforming them into runtime DDIs that are used to dynamically assure CPS safety at runtime.

2.1 What Are Digital Dependability Identities (DDI)?

DDI Definition. A fundamental problem of current dependability engineering processes hampering effective assurance lies in the fact that safety argument models are not formally related to the evidence models supporting the claim. Such evidence models include hazard and safety analysis models and dependability process execution documentation. As such artifacts refer to the same system and therefore are naturally interrelated with each other, we claim this should also be the case for the system's model-based reflection: The *Digital Dependability Identity* (DDI) [2]. By establishing this kind of traceability, DDIs represent an integrated set of dependability data models (i.e. evidence), generated by engineers and reasoned upon in dependability arguments (i.e. how are claims supported by evidence). A DDI is, therefore, an evolution of classical modular dependability assurance models, allowing for comprehensive dependability reasoning by formally integrating several separately defined dependability aspect models. DDIs are produced during design, certified on system/component release, and then maintained over the system/component lifetime.

DDI Contents. A DDI contains information that uniquely describes the dependability characteristics of a system or component. DDIs are formed as modular assurance cases, are composable and can be synthesized to create more complex DDIs from the DDIs of constituent systems and system components. The DDI of a system contains a) claims about the dependability guarantees given by a system to other systems b) supporting evidence for the claims in the form of various models and analyses and c) demands from other connected systems being necessary to support the claims.

Previous Work on DDIs. In the first phase of the DEIS project, the focus has been to integrate various state-of-the-art design-time safety and security engineering aspects together into an exchange format, the *Open Dependability Exchange* Meta-Model (ODE), used as the blueprint for DDIs. Accompanying engineering methods and tools have been developed to enable distributed dependability engineering in multi-tier integrator-supplier scenarios [3]. In [4], we showed, for an industrial railway use case, how to use design-time DDIs to automatically verify safety requirements with component fault trees and model-based evidence lifecycle documentation.

Runtime DDIs. To cope with the openness and adaptivity CPS challenges safely, systems can be engineered in a way that enables them to assure dependability at runtime on their own. Consequently, runtime DDIs need to be developed with appropriate model contents and runtime mechanisms to enable dependable integration and cooperation at runtime. The upcoming sections take the approach outlined in [4] to engineer design-time DDIs as a basis and add on top the aspect of how to address openess and adaptivity

challenges. Therefore, the specific content additions of runtime DDIs are explained, exemplified for a cooperative platooning application and a process for their systematic derivation is proposed.

2.2 Runtime DDI Engineering Approach

Being equipped with knowledge about high-level DDI contents, this section describes from a bird's eye view, how the DDI Dependability Engineering Framework bridges the gap between a CPS use case description and its dependable operation at runtime. Figure 1 visualizes the principal building blocks of CPS dependability assurance.

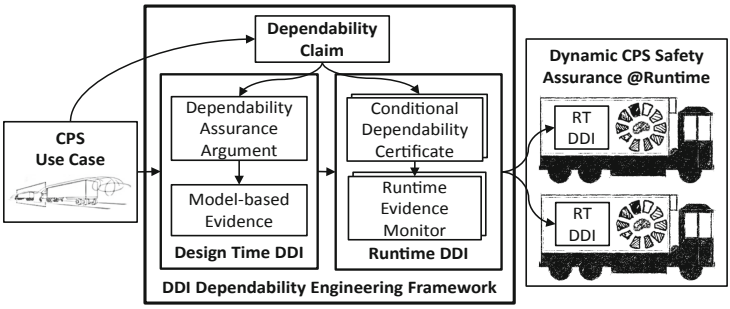

Fig. 1. DDI dependability engineering framework overview.

CPS Functionality. The starting point for all dependability assurance activities is the description and planning of the functionality that the CPS shall render for its stakeholders, which may be either direct system users, companies or even the society. An essential property of a CPS function is that it is executed on multiple independent systems leading to a required distribution of dependability assurance over multiple system manufacturers. For example, a platooning CPS function is executed on multiple trucks of potentially different manufacturers. Enabling cooperative function execution while still allowing decoupled development is only possible by making development and runtime execution interfaces explicit for both functional and quality aspects. Concretely, structural and behavioral aspects of the intended CPS function must be made explicit along with assured constraints regarding their quality bounds.

Dependability Claim. DDIs are concerned with the comprehensive and transparent assurance of dependability claims. Each assurance activity and each artifact contained in a DDI is motivated by a root dependability claim defining risk reduction for a dependability property such as safety, security, availability or reliability. The definition of *acceptable risk reduction* is typically derived from domain-specific risk management standards targeting different risk causes such as functional safety causes (e.g. ISO 26262), causes related to functional insufficiencies and foreseeable misuse (e.g. SOTIF PAS 21448) or causes due to cyber-security threats (e.g. ISO/SAE 21434). These standards contain

requirements for assessing risk criticality and reducing risks to an acceptable level. Note that existing standards do not specifically consider CPS challenges as of now. However, the DDI framework has been defined generally enough to be open for structured extension with contents from future risk management standards specific for CPS assurance challenges.

Design-Time Dependability Assurance. Having a dependability claim to be assured for the CPS function, risk management activities must then be systematically planned. These activities create necessary evidence for supporting the system engineers' reasoning that the dependability claim holds for the developed system/CPS. For both risk management planning and dependability assessment purposes, an explicit argument inductively relates created evidence to the top-level claim through layers of argumentation. While the performed activities and produced artifacts vary depending on the kind of risk that is being managed, argumentation supported by evidence is mandatory for all risks. DDIs deal with dependability risks, thus the currently supported design-time DDI assurance activities and evidence focus on well-established dependability methods such as hazard and risk analysis, safety and security analyses, safety design concepts, validation, and verification. These activities are effective in demonstrating dependability of closed embedded systems, unrelated to the CPS challenges. In addition, reliance on model-based approaches already compensates for the increasing complexity of closed systems. Thus, we believe model-based development is also necessary for assuring CPS.

Runtime Dependability Assurance. The open and adaptive nature of CPS, combined with their increased need for environmental operational awareness to render optimal functionality, increases their complexity tremendously. To assure with sufficient confidence that CPS behavior is dependable in all situations, dependability assessment of those situations is mandatory. A common way to simplify this process is to build the system using worst-case assumptions about the environment, specific for the managed risk. Thus, we only look at the most critical situations and constrain system behavior to be dependable in those situations. The problem with this strategy is that worst-case assumptions lead to performance loss. An alternative to unacceptable performance due to design-time worst-case assumptions is to enable the CPS to reason about dependability at runtime. This alternative involves determining the worst case of the *current* operational situation instead of acting according to the worst case of *all possible* situations. This approach avoids the commonly known state-space explosion problem but demands engineering dependability intelligence into the CPS. Such dependability intelligence builds upon the design-time assurance case by equipping a system with pre-certified knowledge about dependability guarantees it can offer and dependability demands it needs from other systems or the environment to render those guarantees. Additionally, the dependability intelligence needs to monitor both CPS and environment for changes (*Runtime Evidences*) that affect dependability. Based on such changes, it can reason about possible CPS configurations leading to dependable CPS behavior in different situations. Summarizing, runtime DDIs are a reduced form of pre-certified design-time dependability assurance cases, containing only those dependability artifacts and reasoning intelligence required for monitoring dependability-relevant context changes and reacting to them in a dependable way.

Runtime DDI Engineering. Regarding the engineering of concrete runtime DDIs, the DEIS consortium focuses on the usage of *Conditional Safety Certificates* (ConSerts) [1] for expressing modular, variable and fully formalized safety concepts including required runtime evidences enabling safety guarantee-demand matching and thus a basic form of dependability reasoning at runtime. For monitoring CPS state and environment, the consortium explored state-based probabilistic methods such as Bayesian Networks [5]. In Sect. 3, we focus on explaining concretely for the platooning use case, how its design-time DDI looks like and how the ConSert part of runtime DDIs is systematically synthesized. Note that although Sect. 3 demonstrates the usage of DDIs for assuring a *safety* claim, the overall engineering procedure is similar for other dependability properties.

3 Runtime DDI Engineering for Platooning

This section exemplifies the runtime DDI engineering activities and artifacts needed to fulfill the safety claim for a platooning function executed on a CPS.

3.1 Platooning Use Case Description

Function and Constraints. The goal of truck platooning or more general cooperative adaptive cruise control (CACC) is to reduce fuel consumption of all involved vehicles by maintaining reduced air drag at small inter-vehicular distances. Platooning is particularly relevant for heavy-duty trucks due to their high air resistance area and thus hold high potential for fuel economy. Since humans have limited reaction capabilities compared to automated systems, they cannot safely drive at distances where air drag is reduced through slipstream. The example in this paper is limited to automated longitudinal control of a two-vehicle platoon. The platoon's leader truck is assumed to be driven by a human and the follower truck's longitudinal motion is controlled by the platooning system, which is executed in a distributed CPS fashion on both vehicles.

Safe Nominal Behavior. For human-driven vehicles or ADAS, it is often the correct realization of the driver's intent that defines safe behavior for control software. For highly and fully automated systems, this is more complex due to many aspects of the environment that have to be considered in the safe nominal behavior specification. The currently most comprehensive safe nominal behavior definition for highly automated vehicles has been published by Intel/Mobileye in 2017 coined *Responsibility Sensitive Safety (RSS)* [6]. In this paper, we use the RSS formalization of a longitudinal safe distance as safe nominal behavior definition (see Eq. (1)).

$$d_{safe} = \left[v_F \rho + \frac{1}{2} a_{max,acc,F} \rho^2 + \frac{\left(v_F + \rho a_{max,acc,F}\right)^2}{2a_{min,brake,F}} - \frac{v_L^2}{2a_{max,brake,L}} \right]_+ \quad (1)$$

Effectively, the first three terms together represent the stopping distance of the follower vehicle considering a) a reaction distance (v_F: follower speed, ρ: reaction time), b) an acceleration distance if the follower constantly accelerates with $a_{max,acc,F}$ during

reaction time and c) the follower braking distance, when the follower constantly brakes with deceleration $a_{min,brake,F}$. To compute the safe distance, we subtract the leader braking distance with leader speed v_L and constant leader deceleration $a_{max,brake,L}$ from the follower stop distance. Note that the leader vehicle driver's reaction time is not factored in, because, in this time span, the motion state of the platoon does not change and is therefore irrelevant for safety assurance.

Influences on Safe Distance. By looking at Eq. (1), we observe that the safe distance is dynamic, as it depends on the right-hand-side variables. Within the platoon context, such changes can either be triggered in the environment, e.g. road surface conditions may affect the minimum and maximum deceleration capabilities of both trucks. Alternatively, the Wi-Fi communication quality is affected by weather conditions such as precipitation influencing follower reaction time. In contrast, the safe distance may be affected by CPS-internal states such as vehicle mass (very dynamic specifically for trucks) or quality of vehicle speed determination. In summary, the CPS function description together with a safe nominal behavior specification provide sufficient information to start DDI-driven safety assurance.

3.2 Design-Time Safety Engineering with DDIs

This section describes how a platooning system safety assurance case is synthesized for the platooning function and finally captured in a design-time DDI.

CPS Safety. The design-time DDI depicted in Fig. 2 by definition contains a top-level safety claim, for which a safety assurance argument is developed. The argument associates all safety-related activities and their evidence with each other to create justified belief in the validity of the safety claim. Since safety is a system property, the design-time DDI necessarily spans *the whole collaboration space* on which the CPS function is executed (i.e. leader and follower roles). Note that we refer to roles instead of systems here to highlight the fact that at this point, we are dealing with functional entities and not with concrete constituent systems realizing these functional entities. Conceptually, one truck can have the main responsibility for achieving safety in the end, but the top-level safety claim has to be necessarily defined and analyzed for the entire CPS. To illustrate this point, we note that the inter-truck distance as the main safety property to be assured cannot be interpreted for a single truck alone. Rather, a distance can only be defined between multiple objects and therefore is necessarily a property of the object *group* or in the platooning context, the whole platoon. The goal of CPS safety engineering is thus to decompose a CPS safety claim into a set of safety requirements for each cooperation role that allows the cooperative fulfillment of the safety claim.

Hazard and Risk Assessment (HARA). Following the RSS definition of a safe distance, the CPS state that poses safety risk (hazard) is existent if the actual distance is lower than the safe distance. This observation is straight-forward for the hazard in a longitudinal direction. In general, a more systematic hazard and risk assessment (HARA) should be performed for all potential CPS behaviors in the intended operational environment. Given a safe nominal behavior specification, conventional HARA methods are applicable for

CPS. For determining the worst-case criticality of the hazard in all situations, we adopt the ASIL classification from ISO 26262, which yields an ASIL D criticality. Thus, the platoon's safety goal is specified as *"Safe Truck Distance is not violated during platoon driving (ASIL D)"*.

Functional Architecture. After HARA, functional cause-effect relationships that lead to a violation of the safety goal must be analyzed. To obtain sufficient completeness regarding potential violation causes, requires systematic safety analysis using a functional network linking functional blocks and their cause-effect relations. Conventionally, functional models contain data and information flow connectors between sensing, control and actuation functions. In [7], an approach for the service-oriented definition of cause-effect relationships has been presented, which better supports the derivation of modular failure mode interfaces than purely dataflow-oriented models. As we will see, modularity is a necessary property for the derivation of runtime DDIs. Therefore, service-oriented functional networks (e.g. expressed in languages such as SysML) are a suitable basis for CPS safety analysis. The application service of a platoon is *"Safe Platoon Driving with defined performance"*, which uses actuation services like *"Truck Distance Realization"* or functional services such as *"Follower Brake Distance Computation"*, which use sensing services such as *"Follower Speed Provision"*.

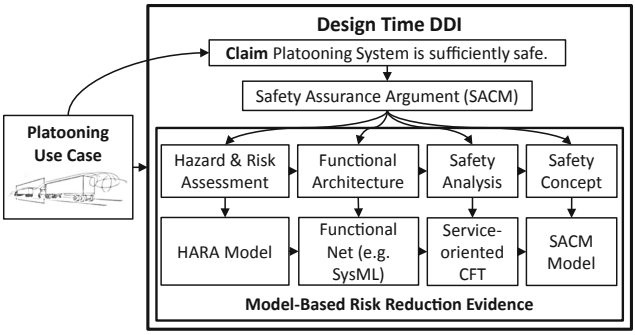

Fig. 2. Contents of a design-time DDI.

Safety Analysis. Starting from a CPS-level hazard, causes leading to this hazard should be systematically identified deductively or inductively. There are several different types of causes to be analyzed, e.g. causes related to a) systematic software faults and random hardware failures, b) functional insufficiencies or foreseeable misuse or c) malicious cyber-security threats. For each cause type, there are specific analysis techniques, e.g. fault trees, failure mode and effect analysis (FMEA), Markov chains or Systems Theoretic Process Analysis (STPA). For platooning safety analysis, we used an extension of component fault trees coined *Service-Oriented Component Fault Trees* (SCFT) [8]. Unlike dataflow-oriented deductive safety analyses starting at the actuators, SCFT analysis starts at the entity with the application context knowledge enabling a derivation of modular interface failure modes. SCFTs build on a stepwise deductive HAZOP guideword interpretation for functional service hierarchies. Example failure modes for a first

analysis step in the context of the platooning safety goal are *"Too Low Computed {reaction, acceleration, braking} Distance"* as these can lead to a violation of the safety goal.

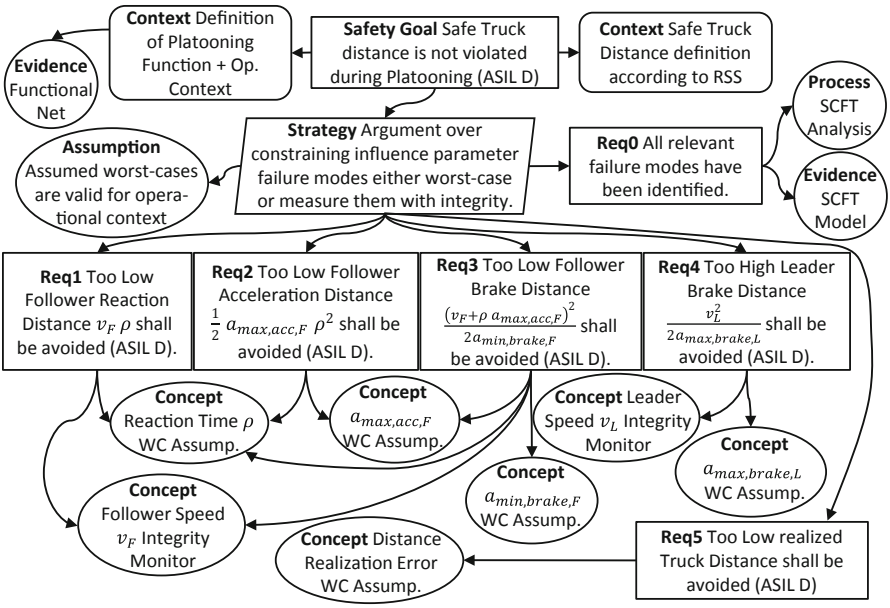

Fig. 3. Platooning functional safety concept.

Safety Concept. After the identification of all functional failure modes, the next step is the creation of a functional safety concept mitigating the propagation from causes to safety goal violation. Figure 3 presents the safety concept documented in an adapted notation based on the Goal Structuring Notation (GSN) [9]. In general, a safety concept provides the rationale for safety measures (depicted as *Concepts*), which add additional design elements for mitigating critical failure modes, i.e. lowering their occurrence probability by failure detection and appropriate transition to a safe state. Note that the *Concept* elements are placeholders for safety argument fragments arguing the requirement satisfaction either through the appropriate choice of worst-case assumptions or dynamic monitoring of measurement integrity. The integrity of safety measure implementation is dependent on the risk criticality of the hazardous event, which originates from the HARA (in our case ASIL D). The safety concept should give a comprehensive argument along with evidence about why a safety goal cannot be violated given the chosen safety measures. In the DDI engineering approach, this evidence is explicit by linking all artifacts such as functional net, the SCFT model along with its analysis results, the definition of the safe nominal behavior and the operational context to the assurance claims they should fulfill (e.g. *Req0*). Together with the evidence artifacts, the safety concept is expressed in a comprehensive safety case representing the design-time DDI's backbone. The DDI formalism for expressing assurance cases is the *Structured Assurance Case Metamodel*

(SACM) [10], which is a successor of GSN and has been recently standardized by the *Object Management Group* (OMG).

CPS Hazard Mitigation Strategies. To derive concrete safety measures to adequately mitigate platooning failure modes (see requirements *Req1–Req5* in Fig. 3) there are two potential strategies: On the one hand, we can postulate and use worst-case assumptions, which must be valid for the intended operational context. For instance, if we assume the maximum leader deceleration to be bound by 9,81 m/s^2 (=1 g), we have to provide an additional argument that justifies the assumption's validity, e.g. through physics reasoning. Worst-case assumptions are a classical means to simplify safety assurance at the cost of under-performance in non-worst cases. For instance, we can consider the larger-than-necessary controlled truck distance for the majority of operation time the lead driver does not perform emergency braking. On the other hand, we can monitor the variables of interest at runtime to replace the worst-case value with the actual runtime value (see Fig. 4, left). For instance, a common established measure in vehicles is speed integrity monitoring. In order to use runtime monitors safely, we have to design them with the same integrity as the hazard they should mitigate. This means, that an assurance argument has to be developed for the correct provision of a "situation-specific worst-case speed value bound", which effectively means that the speed provision error is constrained by employing adequate redundancy mechanisms developed with appropriate safety processes.

In summary, this section led through the systematic engineering of a design-time DDI for the platooning function leading to a justified functional safety concept. The next step is to modularize and abstract this safety concept into runtime DDIs that can be deployed to concrete trucks, enabling dynamic safety assurance for a platoon.

3.3 Runtime Safety Model Derivation

In this section, CPS assurance challenges and potential solutions principles are presented first. Afterward, the design-time DDI of the CPS-based platooning function is systematically transformed into ConSerts, which build the fundamental basis of a runtime DDI suitable for dynamic safety assurance of a CPS at runtime.

CPS Assurance Challenges. The major challenge of safety assurance of CPS functionality is complexity. More specifically, this comparative increase in complexity stems from two CPS-inherent characteristics: open context and adaptivity. *Open context* means that CPS constituent systems are typically developed independently, but should, in the end, provide a common functionality at runtime. Thus, the concrete nature of potential cooperation partners is intentionally left open for the sake of flexibility. In contrast, *adaptivity* means that CPS need to adapt themselves to changing context conditions safely. Although adaptivity was already required for partly automated systems, the amount of required situational awareness has tremendously increased for fully automated CPS functionality. For *safety* assurance, both open context and adaptivity are hard problems, because safety as a property of the entire CPS system *group* needs to be decomposed onto independently developed constituent systems, which finally need to collaboratively adapt safely to changing context.

Open Context Solution. Complexity is not a new phenomenon in the domain of systems and software engineering: A proven solution principle for tackling system-level complexity is *Divide-and-Conquer* (D&C). The idea behind D&C is to iteratively identify a service interface and decompose the solution across its boundary. If such an interface exists, both user and provider of the interface can be developed independently, thereby promoting *design-time modularity* and *runtime compatibility*. To apply D&C to the CPS' open context, we need an explicit and semantically complete interface definition a) for functionality aka *services* being provided or required at that interface and b) for safety guarantees and demands associated to the service interface. In Fig. 3, the mentioned service and safety interfaces emerge immediately as the *Concepts* are deployed to our envisioned systems *Leader Truck* and *Follower Truck* (see Fig. 4, right). The deployment of safety measures is a critical decision with impact, as it determines a) the system role that is finally responsible for achieving the CPS safety at runtime and b) the safety guarantee/demand interface required between trucks, which is directly related to the effort for defining the interface. We found in past projects that minimizing the number of required services and associated safety demands is a good rule of thumb for deployment. In our platoon, the follower truck gets the overall safety responsibility, mainly because it is the only vehicle that can transition the platoon to a safe state in case the communication link is down. We cannot rely on the leader truck's human driver in this regard. In contrast, the leader truck can provide the leader's current speed with much better quality from its internal sensors than measuring it remotely in the follower. In summary, by employing the D&C principle on the design-time DDI, safety concepts can be modularized by deploying the safety measures to the CPS cooperation roles to be implemented by specific systems.

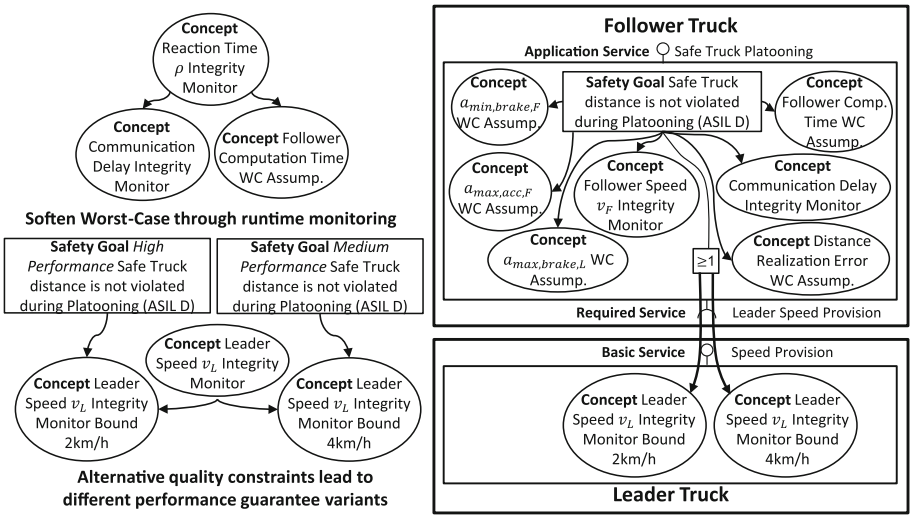

Fig. 4. *Left*: platoon variant analysis, *Right*: modular platoon safety concept.

Context Adaptivity Solution. Cooperation-partner diversity and context-dependent optimal performance necessitates context adaptivity. For instance, different truck models from different brands may have different functional capabilities for sensing and actuating with different safety guarantees for these capabilities. Further, CPS must adapt to changing risk levels as the environment changes. Different risks lead to situation-dependent safety requirements and situation-dependent optimization potential. Without context adaptivity, there would be *exactly* one possibility to cooperate safely. Regarding the environment, this would need to be the set of worst-case conditions for all properties, while for the truck interface exactly one configuration of leader capabilities would be allowed by the follower. Therefore, variants need to be factored into both service/safety interface and safety concept that a) increase chances of interface compatibility and b) enable situation-dependent optimal performance while still maintaining safety. Two types of variants are depicted in Fig. 4, left: The top variant decomposes the reaction time worst-case assumption into another worst-case assumption about follower computation time, but makes the communication delay *dynamic* so that the platoon can adapt to different situations. The other example induces variants on the safety demand bound for the provided leader speed leading to different *performance* guarantees for the overall platoon since the safe distance computation still considers the situation-specific worst-case.

Information Abstraction. The last property that distinguishes a design-time DDI from a runtime DDI is the amount of incorporated information. After modularizing the CPS safety concept into black-box system safety concepts with defined interfaces, each truck manufacturer can certify their implementation including all pre-engineered variants *conditionally* at design-time. Thus, runtime DDIs contain only the information required to reason about variable safety conditions dynamically at runtime. These conditions are a) a representation of the service interface with its safety guarantees and demands, b) a system-internal mapping of safety guarantees and their required safety demands, and c) context variable requirements for monitoring regarding their concrete equivalence classes. In addition to the above, a runtime DDI also requires mechanisms for safety interface matching, variant resolution, and runtime context monitoring.

ConSerts. *Conditional Safety Certificates* (ConSerts) [1] is a concrete instance of a runtime safety assurance approach that unifies all above explained solution building blocks for CPS safety assurance. ConSerts support *open context* in that they use a black-box service architecture to achieve modularity. Through ConSerts, provided and required functional services are enriched with safety guarantees and safety demands defining a) formalized failure modes along with variable bounds to be assured for them, b) context-specific constraints such as situation, for which these bounds have to be valid and c) an integrity statement indicating the confidence required for the assurance of the bounds. ConSerts support *adaptivity* in that they allow to define a) different safety guarantees and demands for a single service and b) monitored context properties to provide *runtime evidences*. ConSerts support *information abstraction* through the usage of Boolean logic for expressing safety guarantee/demand and runtime evidence relationships thus abstracting the design-time safety concept to the mere dependency logic being relevant at runtime.

Platooning Runtime DDIs. Figure 5 shows runtime DDIs for the leader and follower platoon trucks. They contain the service interface, its safety guarantees (SG), safety demands (SD) and required runtime evidences (RtE). The *"Safe Truck Platooning"* service can be guaranteed by the follower truck with high quality (=small truck distance, SG1), if the communication delay does not exceed a certain quality bound and the follower speed can be determined with the required confidence (RtE1,2). In addition, deviations of the provided leader speed bound to max 2 km/h are demanded from the leader truck to fulfill SG1. One potential realization of SG4 is to employ diverse redundancy regarding the measurement principle of the leader's speed (RtE3,4). If RtE 3 cannot be provided anymore due to e.g. ESP system failure, graceful degradation is triggered through ConSert evaluation in that only SG2 can be guaranteed given that RtEs 1, 2 and 4 are still existent. This leads to a softer safety demand to the leader (SD2) yet still better performance than deactivation of the cooperation. In this way, ConSerts allow the definition and dynamic negotiation of arbitrary degradation level variants between the maximum (SG1) and no performance (SG3,6), while always guaranteeing CPS safety in a modular fashion during CPS cooperation.

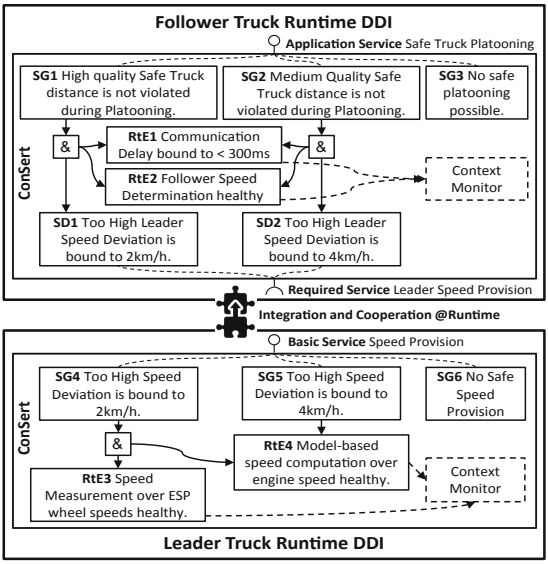

Fig. 5. Platooning runtime DDIs and ConSerts for leader and follower trucks.

4 Related Work

In the SPES projects [11], the Open Safety Metamodel (OSM) enables modular, cross-tool and cross-company safety certification. The OSM had a major influence on the evidence formalization in design-time DDIs. The AMASS project focusses on organizing safety cases, formalized in the Common Assurance and Certification Metamodel

(CACM) [12]. The CACM is an extension of the DDI's Structured Assurance Case Metamodel (SACM), in that it adds capabilities to model risk management standard terminology. Integrating CACM in the DDI is ongoing work to extend the latter with further formalization capabilities regarding concepts and terminology from dependability standards as well as evidence management processes. In contrast, there are pure runtime safety monitor approaches such as [13], which dynamically assess risk at runtime and react appropriately. The DDI improves upon these works by seamlessly integrating design-time safety assurance and runtime monitoring.

5 Conclusion and Future Work

In this paper, we described a continuous engineering method for safety assurance of CPS-based functionality via ConSerts. Our method seamlessly integrates development time and runtime assurance to provide safety claim confidence. The DDI dependability-engineering framework in Sect. 2 unifies development time and runtime assurance artifacts in an integrated data store, the DDI. In Sect. 3, we executed the DDI engineering workflow for a CPS-based platooning function, from use case description over integrated development time assurance (HARA, functional analysis, safety analysis, safety concept) to the derivation of modular safety concepts and ConSerts as fully formal runtime safety models. The demonstration suggests that the DDI framework can bridge the gap between established development time assurance practice and innovative runtime assurance concepts for tackling uncertainties and unknowns of CPS at runtime. In the future, we look to enrich runtime DDIs with probabilistic reasoning schemes to account for uncertain perception at runtime. *Dynamic Safety Management* [14] is a promising conceptual framework for runtime safety assurance, which we see as a roadmap for the evolution of DDIs.

Acknowledgment. This work was funded by the DEIS Project (EC Grant 732242).

References

1. Schneider, D.: Conditional safety certification for open adaptive systems. Dissertation, Technical University of Kaiserslautern, Germany (2014). ISBN 978-3-8396-0690-2
2. Schneider, D., et al.: WAP: digital dependability identities. In: Proceeding of IEEE International Symposium on Software Reliability Engineering (ISSRE), pp. 324–329 (2015)
3. DEIS Consortium: Dependability engineering innovation for cyber-physical systems project dissemination. http://www.deis-project.eu/dissemination/. Accessed 21 May 2019
4. Reich, J., Zeller, M., Schneider, D.: Automated evidence analysis of safety arguments using digital dependability identities. In: Romanovsky, A., Troubitsyna, E., Bitsch, F. (eds.) SAFECOMP 2019. LNCS, vol. 11698, pp. 254–268. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26601-1_18
5. Kabir, S., et al.: A runtime safety analysis concept for open adaptive systems. In: Papadopoulos, Y., Aslansefat, K., Katsaros, P., Bozzano, M. (eds.) IMBSA 2019. LNCS, vol. 11842, pp. 332–346. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-32872-6_22
6. Shalev-Shwartz, S., Shammah, S., Shashua, A.: On a formal model of safe and scalable self-driving cars. Intel/Mobileye (2017). http://arxiv.org/pdf/1708.06374v5

7. Reich, J., Schneider, D.: Towards (semi-)automated synthesis of runtime safety models: a safety-oriented design approach for service architectures of cooperative autonomous systems. In: Proceeding of 13th International Workshop on Dependable Smart Embedded and Cyber-physical Systems and Systems-of-Systems (DECSOS), Västerås, Sweden (2018)
8. Adler, R., Schneider, D., Höfig, K.: Evolution of fault trees from hardware safety analysis to integrated analysis of software-intensive control systems. In: Proceeding of 27th European Safety and Reliability Conference (ESREL), Portoroz, Slovenia (2017)
9. Kelly, T., Weaver, R.: The goal structuring notation - a safety argument notation. In: Proceeding of the Dependable Systems and Networks Workshop (2004)
10. Wei, R., Kelly, T.P., Dai, X., Zhao, S., Hawkins, R.: Model-based system assurance using the structured assurance case metamodel. J. Syst. Softw. **154**, 211–233 (2019). https://doi.org/10.1016/j.jss.2019.05.013
11. Pohl, K., Hönninger, H., Achatz, R., Broy, M. (eds.): Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34614-9
12. de la Vara, J.L., et al.: Model-based specification of safety compliance needs for critical systems: a holistic generic metamodel. Inf. Softw. Technol. **72**, 16–30 (2016)
13. Eggert, J.: Predictive risk estimation for intelligent ADAS functions. In: IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), Qingdao, China (2014)
14. Trapp, M., Schneider, D., Weiss, G.: Towards safety-awareness and dynamic safety management. In: 14th European Dependable Computing Conference (EDCC), Iasi, Romania (2018)