

Summary

Citation for published version (APA):

Kochanthara, S., Rood, N., Saberi, A. K., Cleophas, L., Dajsuren, Y., & van den Brand, M. (2021). Summary: A functional safety assessment method for cooperative automotive architecture. In *ECSA-C 2021: Companion Proceedings of the 15th European Conference on Software Architecture* Article 14 (CEUR Workshop Proceedings; Vol. 2978). CEUR-WS.org. <http://ceur-ws.org/Vol-2978/jf-paper14.pdf>

Document status and date:

Published: 01/01/2021

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Summary: A Functional Safety Assessment Method for Cooperative Automotive Architecture*

Sangeeth Kochanthara, Niels Rood, Arash Khabbaz Saberi, Loek Cleophas, Yanja Dajsuren and Mark van den Brand

Eindhoven University of Technology, The Netherlands

Abstract

The scope of automotive functions has grown from a single vehicle as an entity to multiple vehicles working together as an entity, referred to as cooperative driving. The current automotive safety standard, ISO 26262, is designed for single vehicles. With the increasing number of cooperative driving capable vehicles on the road, it is imperative to systematically assess their architectures' functional safety. Many methods are proposed to assess architectures with respect to different quality attributes in the software architecture domain, but to the best of our knowledge, functional safety assessment of automotive architectures is not explored in the literature. We present a method leveraging existing software architecture and safety engineering research, to check whether the functional safety requirements for cooperative driving scenarios are fulfilled in the technical architecture of a vehicle. We apply our method on a real-life academic prototype for a scenario—platooning—and discuss our insights.

1. Introduction

Traffic congestion was estimated to cost 305 billion dollars in 2017 to traffic participants in the United States of America alone.¹ One potential solution to reduce traffic congestion and related operational costs is *cooperative driving*. Cooperative driving refers to the collective optimization of the traffic participants' behavior by sharing information using wireless communication e.g. peer-to-peer networks or via the cloud. It can improve traffic efficiency, reduces cost, and increases comfort [2]. In 2020 alone, 10 million new vehicles with cooperative driving capabilities were projected to hit the roads, and the safety of these vehicles needs urgent attention.²

Most cooperative driving is achieved by determining a vehicle's behavior for optimal traffic behavior based on information received from other traffic participants. Such optimal behaviors are (partially) achieved using software-controlled steering, acceleration, and braking. Any problem in the software can lead to catastrophic effects to the vehicle and other traffic participants. To avoid this, cooperative driving systems are designed to operate in case of failure or fail safely.

The current guidelines to ensure the safety of automotive systems (and their architecture) are provided by ISO 26262 - a product development standard for the automotive domain [3]. The

ECSA 2021: 15th European Conference on Software Architecture, 13-17 September 2021 (virtual)


*This work is a part of the i-CAVE research programme (14897 P14-18) funded by NWO

Use the original publication when citing this work [1]

✉ s.kochanthara@tue.nl (S. Kochanthara)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹<https://www.smartcitiesdive.com/news/gridlock-woes-traffic-congestion-by-the-numbers/519959/>

²<https://bit.ly/volkswagen-includes-nxp-v2x> and <https://www.sciencedaily.com/releases/2019/05/190519191641.htm>

standard offers methods from the safety engineering domain to identify safety requirements. Any automotive software architecture that fulfills these requirements is deemed safe-by-design.

ISO 26262 neither considers cooperative driving nor prescribes methods for architecture assessment. The standard is designed for single vehicles and does not include a cooperative perspective in which a set of vehicles is seen as a single entity [4]. This can mean that a low-risk safety requirement from a single-vehicle perspective can have catastrophic effects on other cooperating vehicles [2]. To create a functionally safe architecture from a cooperative perspective, existing studies have extended the standard guidelines [5] or presented an architecture framework [2]. Yet checking the safety of software architecture of an existing vehicle for cooperative driving, remains an open question.

The ISO 26262 standard does not prescribe methods to assess the *functional safety* of an automotive architecture. Approaches to assess architectures with respect to quality attributes have emerged in the software architecture domain [6]. However, only some methods are designed for operational quality attributes like performance (in contrast to development quality attributes like maintainability) [6]. To the best of our knowledge, none of these methods are designed to assess the operational quality attribute *functional safety* of automotive systems.

This paper presents a method to assess the functional safety of existing automotive architecture for cooperative driving, by combining methods from the safety engineering and software architecture domains. Our method has two parts: (i) derive Functional Safety Requirements (FSRs) for cooperative driving scenarios; (ii) check whether the (technical) software architecture fulfills the derived functional safety requirements—a blend of methods [7, 8, 6] adapted from the software architecture domain. Our paper primarily focuses on the design phase (concept development phase in ISO 26262) and validation of the resultant requirements in the software architecture in the final product. We also validated our approach via a case study on an academic prototype. This extended summary of [1] summarizes our approach; refer to [1] for more details.

2. Methodology

Our method has two parts: (i) derive FSRs for cooperative driving scenarios, and (ii) check whether the FSRs are fulfilled in the technical software architecture of a vehicle.³ Note that the first part needs only a black box view of individual vehicle functions and their interactions and thus uses the functional architecture view.

Derive FSRs for cooperative driving: We extend the traditional method [3] to derive FSRs, to be executed on the entire cooperative system in parallel, rather than on an individual vehicle (see Figure 1). We first outline the traditional method, followed by our prior work on its extension [5] and our new contribution. For the rest of the paper, we use the term *vehicular perspective* for an individual vehicle as a unit and *cooperative perspective* for a set of vehicles.

Traditionally, FSRs for a vehicular perspective are derived by mapping the safety goals for a vehicle on to the individual components of the vehicle’s functional architecture. This process of mapping—safety analysis—captures information on the malfunctioning of a component that can lead to violation of a safety goal. Safety analysis is performed using a systematic process (like fault tree analysis [9]) that takes two inputs: (i) functional architecture and (ii) safety goals.

³The technical software architecture includes the system’s runtime model and allocation of software to hardware.

Safety goals are derived from hazardous events [3], found by decomposing the scenario description using the hazard analysis and risk assessment technique [3]. This method to derive FSRs is depicted by part 3 and flows *a* and *d* of Figure 1, with *a* and *d* as inputs to safety analysis. This method to derive FSRs is the norm in the automotive domain [3].

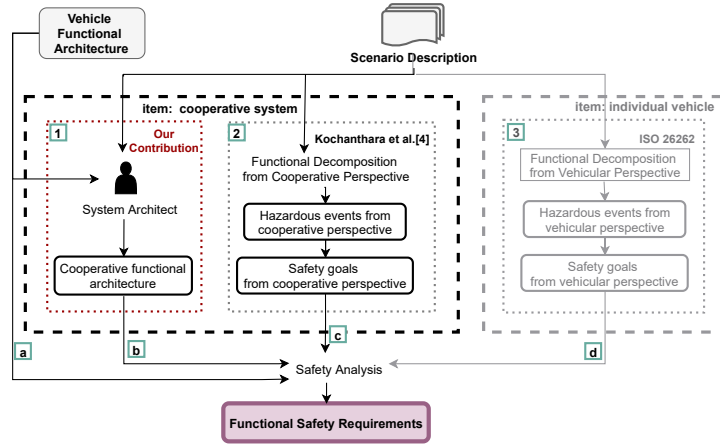


Figure 1: Method to derive FSRs for cooperative driving. Parts 1&2 are our addition. System architect: external entities creating the cooperative architecture.

In the proposed method, we have one item⁴ per individual vehicle type, and an item for the entire cooperative system of which the vehicles are part. A cooperative system can contain more than one type of vehicle (for example, two vehicles with different functional architectures). In the case of more than one type of vehicle, each type will form an item. For the rest of this section, we consider two items: an individual vehicle (representing all vehicle functional architectures) and the cooperative system.

We propose that FSRs for a cooperative system are derived from: (i) safety goals from the vehicular perspective (as in the traditional method), and (ii) safety goals from the cooperative perspective. Our prior work [5] extended the traditional process to derive safety goals for the vehicular perspective to the cooperative perspective (annotated as part 2 in Figure 1) to cover safety goals from both perspectives. This process partitions the scenario description into vehicle-specific and cooperation-specific parts. We apply the traditional safety goal identification steps to the two parts. FSRs from the vehicular perspective are then derived.

A cooperative functional architecture is required to derive FSRs, by mapping safety goals to functional architecture components. The cooperative functional architecture should be built using individual vehicle functional components to preserve the mapping between functional architecture of cooperative system and the technical software architecture of individual vehicles. We propose that the cooperative functional architecture be built from (i) the functional architecture of individual vehicles that constitute the cooperative system and (ii) the cooperative scenario description of the interaction between individual vehicles. With these requirements, system architects can create a functional architecture of the cooperative system such that the individual components of the architecture are mapped onto the components of the functional architecture of vehicles. This process is labeled as part 1 in Figure 1; the complete process of deriving FSRs from the cooperative perspective is shown by the labels 1, 2, *b*, and *c*.

We performed a case study on an academic prototype for the cooperative driving scenario, *platooning*, in which a manually driven vehicle is autonomously followed by a train of vehicles. Application of our method resulted in an additional 9 safety goals and 15 FSRs to the 16 safety goals and 16 FSRs from the traditional ISO 26262 method [1].

⁴Item: “system or combination of systems, to which ISO 26262 is applied, that implements a function or part of a function” [3]

Check fulfillment of FSRs: Our method of assessing the fulfillment of FSRs is a blend of techniques adapted mainly from the software architecture domain. With no existing architecture assessment techniques addressing functional safety in the context of automotive systems, our method takes inspiration from traditional architecture assessment techniques like ATAM [6] and uses the safety tactic framework [7, 8] to leverage existing architecture knowledge.

Our method to check for the fulfillment of FSRs in the technical software architecture of individual vehicles is organized in two phases. Phase one ensures that it is possible to realize all the FSRs by identifying whether there are conflicting FSRs. Phase two describes a systematic method to check for the fulfillment of FSRs in the technical architecture (see figure 2).

In phase one (see Figure 2), we check for conflicting FSRs. Two FSRs are conflicting if both of them cannot be fulfilled at the same time. Comparing every pair of FSRs for conflicts will lead to a quadratic number of comparisons (if n is the number of FSRs, the number of comparisons is $n(n-1)/2 \approx O(n^2)$). We compare FSRs that belong to the same functional architecture component for conflicts. This can reduce the number of comparisons up to a factor of d , where d is the number of functional components (i.e., the number of comparisons can be reduced up to $n(n-d)/d \approx \Omega(n^2/d)$). The reduction is possible since safety analysis techniques for deriving FSRs ensure that each FSR belongs to only one functional component [9, 3]. FSRs belonging to a component can have conflicts among themselves but not with the FSRs belonging to other components.

An FSR may be fulfilled by a safety tactic or a combination of safety tactics. To identify whether an FSR is fulfilled, we propose checking the vehicle technical software architecture for the implementation of safety tactics [8, 7] that can meet the FSR. This is achieved in two steps: (i) identify a set of safety tactics (hereafter referred to as *applicable tactics*) such that the implementation of each tactic, in itself or in combination with some other tactics in the set, can fulfill the FSR; and (ii) check whether any feasible combination of tactics from the applicable tactics that are present in the vehicle technical architecture meets the FSR. Note that, for an FSR f_i and its corresponding functional component c_i , the applicable tactics for f_i need to be compared with only the safety tactics implementations used in the technical architecture counter part of c_i and its associated safety mechanisms, since f_i is only associated with c_i .

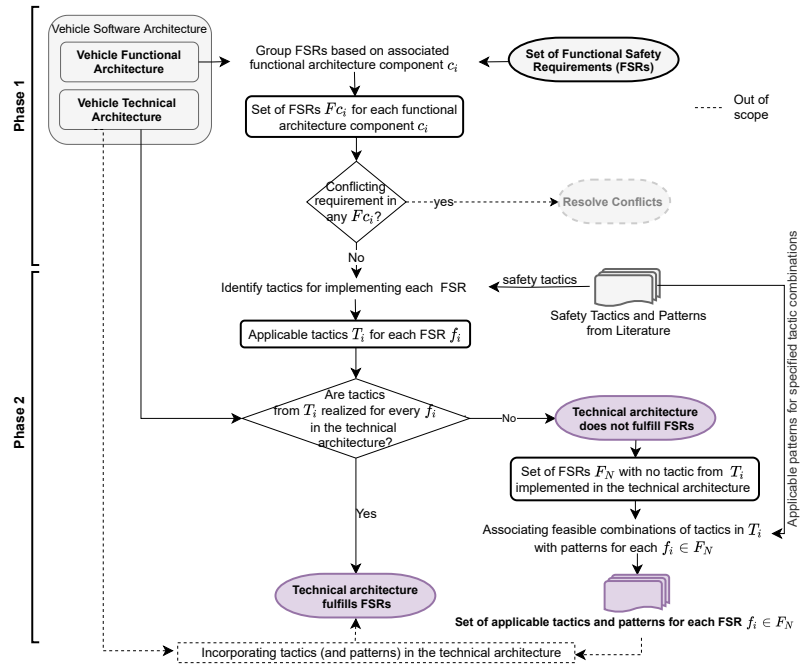


Figure 2: Method to check the fulfillment of FSRs in technical architecture

Applicable tactics for an FSR can be identified based on the FSR description (by navigation of a tactic hierarchy) [6, 8, 7] or by matching the FSR description to the descriptions of tactics [8].

By the end of this two step process of identifying applicable tactics and checking the technical architecture for them, we will have a list of FSRs that do not have any feasible combination of tactics implemented. The list shows the FSRs that have not been fulfilled by the technical architecture, if any. As a by-product, for each unfulfilled FSR, we will also have a set of applicable tactics such that some feasible combinations from this set can fulfill the FSR. These combinations point to a set of safety patterns since safety patterns are associated with the safety tactics they implement [8]. These applicable safety patterns (and applicable tactics) provide the system architects with a set of possible design decisions to realize the unfulfilled FSRs.

In our case study, we checked whether the FSRs (the 31 FSRs as mentioned in Section 2) are fulfilled on an academic prototype capable of cooperative driving using 13 most used safety tactics [8, 1]. We found that the prototype's technical architecture meets only 6 FSRs [1].

3. Conclusion

This paper investigated whether the architecture of a single vehicle meets the functional safety requirements for cooperative driving. We proposed a method to ensure that an automotive architecture is functionally safe to operate in given scenarios. The proposed method derives FSRs for a cooperative driving scenario and checks whether they are fulfilled in the technical architecture of a vehicle. The method is a combination of methods adapted from the safety engineering and software architecture domains. We showed the usability of our method for a cooperative driving scenario—platooning—on a real-life academic prototype, which resulted in uncovering FSRs that were not fulfilled by its software architecture. Our method is motivated by and reinforces the notion that functional safety should not be an afterthought in the design of automotive architectures, but be used for defining the architecture of the automotive system.

References

- [1] S. Kochanthara, N. Rood, A. Khabbaz Saberi, L. Cleophas, Y. Dajsuren, M. van den Brand, A functional safety assessment method for cooperative automotive architecture, (JSS) (2021). URL: [https://authors.elsevier.com/sd/article/S0164-1212\(21\)00088-1](https://authors.elsevier.com/sd/article/S0164-1212(21)00088-1).
- [2] P. Pelliccione, E. Knauss, S. M. Ågren, R. Heldal, C. Bergenhem, A. Vinel, O. Brunnegård, Beyond connected cars: A systems of systems perspective, (SCP) (2020).
- [3] ISO, ISO 26262: 2018 - Road vehicles – Functional safety, Standard, International Organization for Standardization, 2018.
- [4] Y. Dajsuren, M. van den Brand, Automotive Systems and Software Engineering, Springer, 2019.
- [5] S. Kochanthara, N. Rood, L. Cleophas, Y. Dajsuren, M. van den Brand, Semi-automatic architectural suggestions for the functional safety of cooperative driving systems, in: (ICSA-C), IEEE, 2020.
- [6] L. Bass, P. Clements, R. Kazman, Software architecture in practice, Addison-Wesley, 2012.
- [7] W. Wu, T. Kelly, Safety tactics for software architecture design, in: (COMPSAC), IEEE, 2004.
- [8] C. Preschern, N. Kajtazovic, C. Kreiner, Building a safety architecture pattern system, EuroPLoP '13, ACM, 2015.
- [9] W.-S. Lee, D. L. Grosh, F. A. Tillman, C. H. Lie, Fault tree analysis, methods, and applications a review, IEEE transactions on reliability (1985).