System Health Indicators in Mixed Criticality E/E Systems in Automated Driving Context

Friederike Dollinger[1]✉[0000−0002−4898−2483], Rinat Asmus[3][0000−0002−1288−278X], and Marc Dreiser[2][0000−0002−0621−6422]

[1] Technical University Munich, Boltzmannstrasse 3, 85748 Garching, Germany
friederike.dollinger@tum.de
[2] Fraunhofer Institute for Cognitive Systems IKS, Hansastrae 32, 80686 Munich, Germany marc.dreiser@iks.fraunhofer.de
[3] BMW Group, 80788 Munich, Germany
rinat.asmus@bmw.de

Abstract. One problem standing in the way of fully automated vehicles is the question of how to ensure vehicle safety and the safety of all traffic participants. Standards like ISO 26262 and ISO/PAS 21448 tackle those issues from different viewpoints by defining safety measures and mechanisms. While ISO 26262 focuses on safety hazards arising from malfunctioning of E/E systems, ISO/PAS 21448 stresses hazards due to technological limitations. However, it is an open challenge how system-wide safety can be monitored and validated at run-time. To complement those safety specifications we propose a system-wide run-time safety analysis. Our System Health Management concept is based on so-called Health Indicators (HIs) to propagate knowledge about detected errors and trigger appropriate error reactions. We analyze probable information sources to define meaningful HIs in automated driving context and investigate influence factors, of both ISO 26262 and ISO/PAS 21448. We apply our approach to a case study demonstrating its applicability in an automated driving scenario.

Keywords: Safety · Automated driving system · Health Indicator.

1 Introduction

To classify different levels of automation, SAE International released the standard J3016 defining "Levels of Driving Automation" [4]. Current cars realize levels 0 up to level 2, which offer driver support features. For instance, level 2 features can automatically accelerate and decelerate the vehicle. However, the driver must always supervise and take over in case of critical situations, ergo he serves as a safe fallback state. Transitioning from level 2 to level 3 increases system complexity as the vehicle operates autonomously in dedicated Operational Design Domains (ODDs) without requiring driver supervision. Only after notification, the driver must take over within a specified time span. Level 4 can master all driving tasks within the ODD; the automated driving system of level 5

vehicles is capable to handle all environmental conditions. ISO 26262 [2] provides necessary processes and mechanisms for realizing functional safety by avoiding design faults and offering error detection and handling mechanisms, among other things. Nonetheless, these mechanisms do not automatically account for the absence of functional insufficiencies. To close this gap, ISO/PAS 21448 [3] defines the term 'Safety of the Intended Functionality (SOTIF)' as absence of unreasonable risk due to hazards caused by performance limitations of the intended behavior or by reasonably misuse by the user. And yet, the safety standards only support vehicle development up to automation level 2 as many open issues, such as how to validate and verify functional safety, remain [11]. The diversity of error sources like random hardware faults or software design faults gives an indication of why designing level 3 systems is so challenging. Furthermore, the causal safety chain becomes a system-wide property as the failure of one element impacts the rest of the system. For instance, sensor limitations could lead to erroneous environment perception and dangerous driving maneuvers. These safety violations must be identified at run-time in order to trigger respective mitigation actions. To tackle this problem, we present an approach for system-wide safety monitoring at run-time.
This paper is structured as follows. In Section 2 we present a detailed description of the proposed concept. Section 3 provides an example of how this concept can be applied in an automated driving context. In Section 4 we compare our concept to related work. At the end we give a conclusion and outlook.

2 Proposed Approach

Introducing the notion of Health Indicators shall increase system safety by enabling system degradation and Quality of Service (QoS) mechanisms. HIs on system-level enable system-wide recovery and degradation. Especially automated driving systems rely on mechanisms of redundancy and diversity to improve system safety. Thus, a standardized approach for degradation, not locally but on system level, has to be defined. In case of failure, this allows switching to hot-standby instances. Moreover, the principal idea of QoS is adopted to a safety perspective. Attaching QoS values containing HIs permits service receivers to instantly decide how far they trust the received service and how to process this data. Current vehicle architectures integrate diverse platforms like AUTOSAR or Genivi confirming to different safety and criticality levels. The QoS approach facilitates coordinating safety mechanisms beyond the borders of single platforms and standards.

2.1 Meta-Model

We introduce a System Health Management concept to target system-wide run-time safety analysis. Figure 1 depicts a meta-model of the considered system. Its goal is to formally express vehicle abstraction levels and system properties used for run-time health analysis. This model enables partitioning the vehicle

into different domains that group dependent features. According to safety analysis conducted for ISO 26262 and ISO/PAS 21448 all functional features and domains must have well-defined safety properties. These properties can include redundancy and configuration information for the purpose of analyzing valid configurations and possible degradation strategies. At domain level, those requirements result in a rule set for functional degradation rules. Depending on available features, this rule set allows continued system operation despite failures by reducing the set of active functional features (graceful degradation). For deciding which features to terminate, functional features have an assigned priority. In case of failure, features can implement adaptation strategies to save resources by performance degradation. Functional features are composed of different subsystems, defined by hardware, software, redundancy, and capability properties. The performance degradation rules are based on the subsystem's availability and performance. Our concept considers the described properties of the meta-

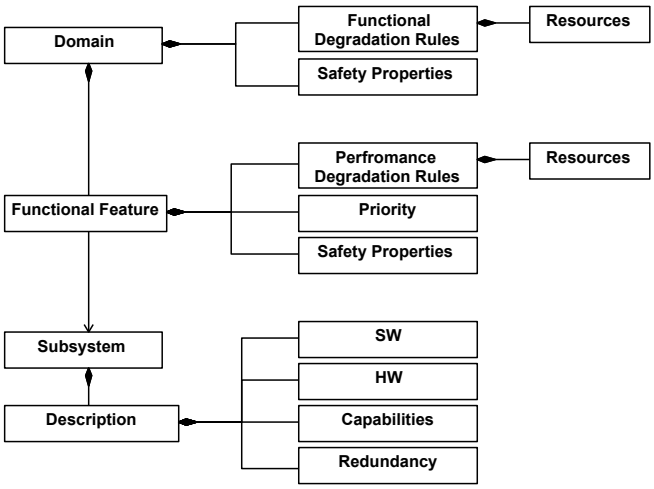

Fig. 1. Meta-model for considered vehicle

model as basis to define constraints and models for run-time Health Indicators. Thus, run-time monitors check those properties at subsystem, functional feature, and domain level. Those monitors are tailored to their underlying systems. Subsystems for example have monitors that supervise critical hardware and software resources. Domains or functional feature monitors may check degradation and availability properties of platforms. Run-time Health Indicators shall enable run-time system degradation strategies. Possible structures of health models are defined in section 3.

2.2 Health Indicator

Within our concept we define Health Indicators as a triple of Performance, Reliability and Degradation: HI = (Per, Rel, Deg). The three parameters capture different aspects required by different safety standards. The Degradation parameter is a system specific set of degradation levels, which are based on availability requirements. ISO 26262 demands supervision and monitoring functionalities to assess the health state of E/E systems with respect to random or system errors. The health state of ISO 26262 related safety considerations is captured in the Performance parameter. The Reliability parameter evaluates how much to trust the system due to uncertainties. Therefore, it encompasses SOTIF related safety considerations by including the vehicle's interaction with its environment, users, and other cars to capture uncertainties introduced by them.
The main purpose of Health Indicators is to monitor system-wide safety properties at run-time in order to trigger appropriate mitigation actions. Safety violations can be mitigated by reducing the system's functionality or performance. Our Health Indicator triple supports both degradation strategies. The Degradation parameter gives an overview on available system resources for functional degradation. The Performance and Reliability parameters of features or domains are a valuable information source to trigger performance degradation strategies.

2.3 Run-time System Health Management

The self-adaptation strategy of the SHM is implemented as MAPE-K loop. It refers to the five activities of **M**onitoring the environment and/ or system, **A**nalyzing data for discrepancies, **P**lanning possible adaptation strategies, and finally **E**xecuting the adaptation based on modeled **K**nowledge [10]. Figure 2

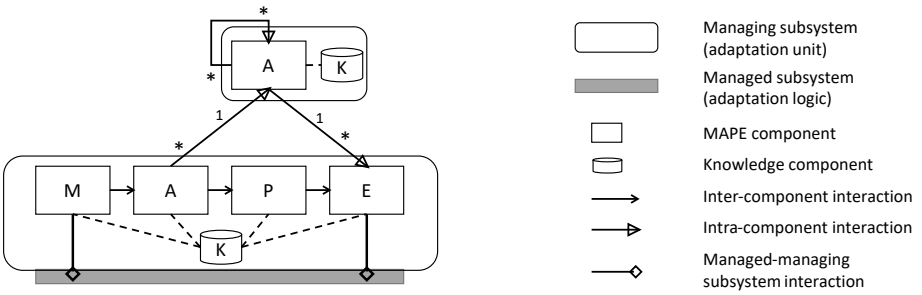

Fig. 2. Adaptation mechanism structure

illustrates the planned MAPE-K adaptation architecture. Each managed subsystem consists of a local MAPE-K loop. Subsystems can be single software platforms, ECUs, or sensors, for instance. Relevant health information is locally collected during the analysis phase and is shared with a global analysis unit,

the System Health Manager (SHM). The SHM receives information from one or multiple local analysis units and optionally also from other SHMs. Based on this information, HIs on subsystem, functional feature or domain level are determined. The HIs are in turn shared with managed subsystems. For clean architectural structuring, the concept shall comply to the "separation of concerns" principle. Therefore, the adaption logic and execution are left with local state managers, as they are the only ones with detailed information on, for instance, running processes. The SHM focuses on abstract global health analysis and provides this information via HIs to local managers. Extensive system analysis is required to ensure the local adaptations lead to a globally consistent state as for instance presented in [5]. For safe global recovery, it may be necessary for dependent subsystems to exchange current states.
As a standalone solution, our concept cannot guarantee safe system behavior but rather acts as one measure to enable system degradation strategies by providing self-awareness on the system's health status. For example, one SHM is a single point of failure and might violate safety requirements. To circumvent this problem, two redundant global SHMs can be implemented. The second SHM serves as hot-standby instance and can take over in case of failure of the first instance. Therefore, compliance with safety requirements is dependent on combining safety mechanisms tailored to the underlying problem statement.

3 Health Indicators in Automated Driving Context

3.1 Use Case

This section presents an example for determining HIs on subsystem level. The paper is taken from the thesis [6], refer to it for more refined Health Indicator examples. Figure 3 shows a simplified logical architecture of the Automated Driving domain adapted from [1]. The system enables level 3 features like a "highway pilot", to autonomously navigate vehicles on highways with structurally separated roads with a maximum velocity of 130 km/h. In case of severe system failures, the vehicle can either continue its operation with a reduced velocity of 60 km/h, request the driver to take over after a specified time, or stop the vehicle at the emergency lane. To define HIs the presented E/E system architecture is analyzed and divided into subsystems. In the following, the nominal integration platform is taken as an exemplary subsystem for calculating HIs. The nominal integration system is responsible for trajectory calculation. The computer vision platform provides information to the PAC and the SAC ECUs. Both channels generate independent environment models. Based on this environment model, the PAC application computes a collision-free vehicle trajectory. In parallel, SAC also uses computer vision information to determine a minimal risk trajectory. Afterwards, the PAC Validator and the SAC Validator each check both trajectories for collisions. According to several performance and safety requirements, the trajectories are associated with a score. The Selector uses those scores to choose the best trajectory.

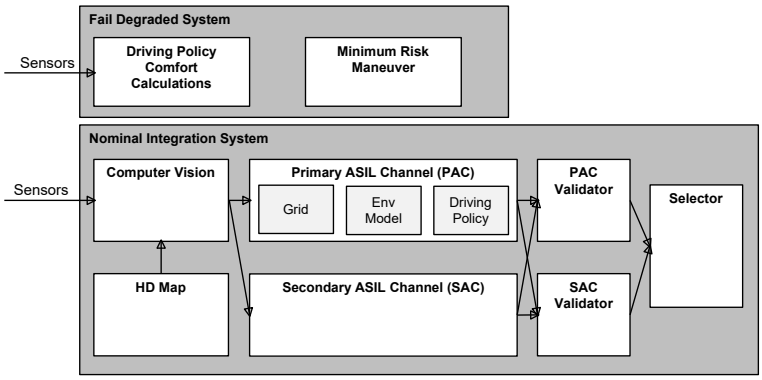

Fig. 3. Example logical architecture for Automated Driving domain

3.2 Health Indicators of Automated Driving System

The Degradation represents different levels based on the availability and causal dependencies of hardware and software resources. PAC and SAC have different purposes; PAC shall continuously operate and compute trajectories while SAC is supposed to take over in case of PAC failure. SAC is not intended for continuous operation. As soon as SAC is activated a handover to the driver is initiated. Those differences shall be visible in different degradation states:

$$Deg_{Nom} = \begin{cases} 0 & \text{Ok} \\ 1 & \text{Minimal Risk} \\ 2 & \text{Failed} \end{cases} \tag{1}$$

For determining the Performance parameter, we propose a rule-based approach based on software monitor results. The nominal integration system is considered safety-critical and different software monitors supervise the timely arrival of sensor information and whether logical and deadline constraints are satisfied. An error tolerance for failed reference cycles can be configured for Alive Supervisions. For instance, all supervision results are summed up in a supervision status, which can have one of four states:

- OK: No supervision failed.
- FAILED: An Alive Supervision failed and the error counter is below the configured error tolerance.
- EXPIRED: A Deadline or Logical Supervision failed or the error counter is equal or above the configured error tolerance.
- DEACTIVATED: A mode switch deactivated the Supervised Entity.

Those supervision states can be mapped to performance levels as shown in Equation 2. OK and DEACTIVATED suggest application performance is good (0). A delayed sensor input might decrease overall performance of the highway pilot regarding its driving smoothness but is not considered a safety risk. Thus, FAILED is in this case mapped to medium performance (1). EXPIRED indicates a severe error or even functionality loss and is mapped to poor performance (2).

$$Per_{Sen} = \begin{cases} 0 & \text{if } LSS = OK \vee LSS = DEACTIVATED \\ 1 & \text{if } LSS = FAILED \\ 2 & \text{if } LSS = EXPIRED \end{cases} \tag{2}$$

The nominal integration system is a poster example of uncertainty introduced by SOTIF for the Reliability parameter. Using machine learning algorithms in computer vision and grid fusion yields high uncertainties as unfamiliar environments might be interpreted the wrong way. The probability values of both machine learning algorithms are indicators of how good the reliability of the environment model is. In addition, the cross-validation of both trajectories evaluates how much the computed trajectories can be trusted. The reliability is influenced by whether the validators agree on the trajectory score and by the number of the score itself. Disagreeing should decrease reliability, agreeing should increase its values. Equation 1 considers the elaborated influence aspects.

$$Rel_{Nom} = (\alpha * Prob + \beta * \frac{\#agree}{\#disagree}) * \frac{score}{score_{max}} \tag{3}$$

3.3 Health Indicator Models

In the following, we outline general aspects for determining Health Indicators on subsystem, feature and domain level. For Degradation and Performance rule-based health models prove valuable. They can encompass supervision results of already installed monitoring mechanisms of safety-relevant software and hardware components like software supervision results or sensor measurements. Possible subsystem failure dependencies are included by inspecting fault trees. The uncertainty mirrored by the Reliability cannot be categorized in concrete states. Instead we propose numerical evaluation with values ranging from 0 to 100. This way, different influence factors can be weighted and put into relation with each other. In general exist two main uncertainty causes: aleatoric and epistemic uncertainty [9]. Epistemic uncertainty covers the uncertainty of unknown situations as there is no guarantee the artificial intelligence system will react safely. Aleatoric uncertainty is caused by inaccurate sensor measurements and results in a misrepresentation of the actual environment. To grasp aleatoric uncertainty, it is important to capture the current environment state as well as the capabilities and availability of different sensors. For measuring epistemic uncertainty diverse and redundant information is used and compared as presented in the example with PAC and SAC. Therefore, results of different validator and plausibility checkers can be weighted to measure Reliability. Another uncertainty factor are

human operators. Consequentially, results of driver monitoring systems are a good information source on the current driver state and could be included in the Reliability analysis.

4 Related Work

Most self-adaptive strategies in the industry and the scientific community focus on application- or component-level analysis. In the following we compare our concept with three system-level solutions. In the avionics domain, [7] presents a two-level approach for software health management. The authors suggest combining Component-level Health Managers with a high-level System Health Manager. Similar to our approach, the Component-Level Health Manager is responsible for monitoring subsystems and reporting the anomalies to the System-Level Health Manager. The System-Level Health Manager conducts a system-level analysis on anomalies and executed mitigation actions. In contrast to our runtime mitigation strategies based on HIs, the overall system diagnosis identifies the root cause and an appropriate coping strategy is selected. Afterwards, the Component-Level Health Manager is informed about the strategy and executes the mitigation actions [7]. This work does not define any abstraction levels for system health as we do with subsystem, features and domains.
In the automotive domain, the "SafeAdapt - Safe Adaptive Software for Fully Electric Vehicles" project presents a decentralized concept for safe runtime reconfiguration. All core nodes cyclically exchange health states of running applications. The knowledge of application health states is used for coordinated global adaptions [12]. We propose collecting health information to determine HIs on feature and domain level in a centralized instance. Our SHM only provides relevant HIs to local instances, which execute the adaptations.
Frtunikj proposes in [8] a decentralized fault management layer to handle system failures. Each local node collects health information to determine a health state of multiple system functions. Those health states are cyclically exchanged. Taking the health state and additional information like redundancy types of a subsystem, its degradation level as well as the system function degradation level is calculated. On this basis the system reconfiguration manager chooses an appropriate adaption option [8]. The proposed system functions are similar to the features of this concept. We further consider the domain level for HIs. Additionally, the HIs do not only give an indication on the current degradation state but include knowledge on Performance and Reliability states.

5 Conclusion and Outlook

To complement existing standards, we set up a generic System Health Management concept for run-time safety analysis based on Health Indicators. Moreover, HIs can help tackling other open challenges for automated vehicles. To bring automated driving into practice without compromising safety requirements, sophisticated verification and validation methods are required. It is yet an unsolved

problem how to generate test scenarios and test data that capture all relevant hazards. Virtual simulation environments are a promising strategy and could further be used for validating our concept in a risk-free environment. But this is only part of the solution. Bringing verification and testing to run-time appears a valid approach to complement the vehicle's safety argumentation. HIs can be seen as a first step in the direction of run-time verification. Our concept currently does not consider security issues. Information corruption of sources like backend servers, other vehicles, infrastructure, or apps would result in severe hazards. Including a run-time security evaluation is as important as the run-time safety assessment. Future work could extend the existing HI with security parameters that indicate external intrusions.

6 Acknowledgments

This work was partially supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as Fraunhofer High Performance Center Secure Intelligent Systems.
The final authenticated version is available online at `https://doi.org/10.1007/978-3-030-59155-7_36`

References

1. BMW Group Safety Assessment Report - SAE Level 3 Automated Driving System. `https://www.bmwusa.com/content/dam/bmwusa/innovation-campaign/autonomous/BMW-Safety-Assessment-Report.pdf`, Accessed: 2020-05-11
2. ISO 26262 - Road Vehicles - Functional Safety (Dec 2018)
3. ISO/PAS 21448 - Road vehicles Safety of the intended functionality (Jan 2019)
4. SAE Standard J3016 - Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (Rev Jun 2018)
5. Becker, K.: Software Deployment Analysis for Mixed Reliability Automotive Systems. Dissertation, Technical University Munich, Munich (2017)
6. Dollinger, F.: Definition of System Health Indicators in Mixed Criticality E/E Systems in Automated Driving Context. Master's thesis, Technical University Munich, Munich (2020)
7. Dubey, A., Karsai, G., Mahadevan, N.: Model-based software health management for real-time systems. In: 2011 Aerospace Conference. pp. 1–18 (Mar 2011). https://doi.org/10.1109/AERO.2011.5747559
8. Frtunikj, J.: Safety Framework and Platform for Functions of Future Automotive E/E Systems. Dissertation, Technical University Munich (2016)
9. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction. ArXiv **abs/1910.09457** (2019)
10. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003). https://doi.org/10.1109/MC.2003.1160055
11. Kirovskii, O.M., Gorelov, V.A.: Driver assistance systems: analysis, tests and the safety case. ISO 26262 and ISO/PAS 21448. IOP Conference Series: Materials Science and Engineering **534** (Jun 2019). https://doi.org/10.1088/1757-899x/534/1/012019

12. Weiss, G., Schleiss, P., Drabek, C.: Towards flexible and dependable E/E-architectures for future vehicles (Sep 2016)