Safety Architectures for Self-Driving Cars: Why and How

Wilfried Steiner
TTTech Computertechnik AG
wilfried.steiner@tttech.com

***Abstract*—Safe self-driving cars must be fail-operational systems. This is a non-trivial problem and a safety architecture guides the system design.**

I. Introduction

In an SAE L4 automated-driving system, the driver would trust the vehicle to operate safely and may, consequently, pay no attention to the driving situation. The driver might text, work, or even sleep. This trust of a human into a machine must be met by the design of the L4 automated driving system as an ultra-high reliable system, i.e., this system may not fail more frequently than, say, 10^-8 failures per hour, which is about once every ten-thousand years. Such low failure rate ensures that there is a high probability that a driver will never experience a system failure of the automated driving system. Achieving this is a challenge, because components such as hardware, chips, and the software that are the building blocks of these systems, fail much more frequently. However, there is a rich body of engineering practice and scientific literature on how to construct ultra-high reliable systems from less reliable components. For example, flight control systems in aerospace are required to fail even less frequently.

Ultra-high reliable systems can only be constructed by means of appropriate decomposition of the overall system into subsystems and fault-containment units with a sufficient level of redundancy. Today's automotive E/E architectures often do not follow such a systematic top-down design approach for ultra-high reliable systems but are rather driven by the decomposition into functional domains. Subsystems and fault-containment units are often retrofitted into a finished system decomposition in a bottom-up manner. This latter approach bears high risks that the reliability goals cannot be met, and *architectural band-aids* must be introduced which typically result in dangerous emerging behavior as a side effect. In this paper we will review design challenges and provide a conceptual architecture for ultra-high reliable L4 automated driving systems.

II. The purpose of a Safety Architecture

Safety, as well as security, is a property of the overall system and it is binary: an autonomous car is acceptably safe, or it is not. When we develop the system, this safety property translates into top-level requirements that guide the system implementation. This is typically a stepwise process in which multiple layers of requirements and designs are derived and we can always verify and validate, step by step, that a design and implementation satisfies the respective requirements. This requirements-driven system development process has been proven to result in safe systems in adjacent industries, like aerospace, but it is also a common approach in the automotive domain. If safety was considered only late in the system development, safety would be introduced retrospectively by adding safety band-aid over safety band-aid with the risk that major safety gaps remain, and adequate system safety may not be achieved.

It is a key requirement, that an autonomous car must continue to operate safely even when some part of it fails. This system property is called *fail-operational*. Designing such a fail-operational system is immensely difficult because there are many different parts that may fail in many different ways. Thus, the number of things that may go wrong is enormous. The only chance to manage this enormous failure space is to adhere to a safety architecture. This safety architecture defines so called fault-containment units (FCUs), i.e., parts of the system that may fail as a whole, and also defines the interactions between these FCUs. The right safety architecture can already ensure, on this level of abstraction, that the failure of any FCU will not cause a complete system failure, but the system will remain operational.

Fail-operational systems are common in the aerospace domain which allows us to learn from aerospace about possible adequate safety architectures. However, the functional complexity of an autonomous car is unprecedented also in the aerospace domain. Even though real autopilot systems do exist in aerospace, self-driving cars have a much more complex task to solve, simply because of the many more dynamic objects one finds on a street rather than in the sky. Moreover, autopilot systems that are used in aerospace must be monitored by trained pilots, which means that these systems typically do not exceed the comparable level 2 in automated driving. On the positive side since the self-driving car is operating on the ground it can quickly enter a safe state once a failure is detected. For example, the car can drive to the emergence lane.

III. A Safety Architecture Proposal for Self-Driving Cars

Kopetz has proposed a safety architecture for self-driving cars in [1] depicted in Figure 1. Automated driving systems that follow this architecture can safely replace the human

978-3-9819263-6-1/DATE22/©2022 EDAA

driver. Quite literary, these systems generate output that otherwise would be produced by the human driver, essentially the setpoints for speed and direction of the car. Even better, these systems will continue to operate even in presence of failures.

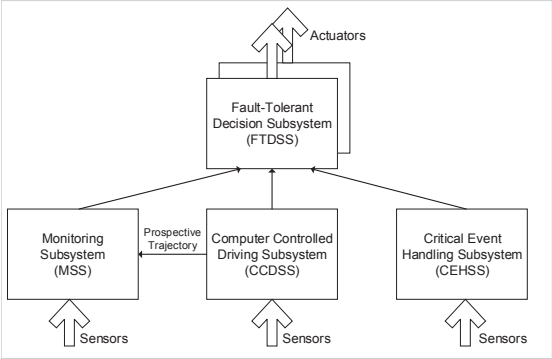

Fig. 1: Safety Architecture for Self-Driving Cars

The architecture distinguishes four subsystems:

- Computer Controlled Driving Subsystem (CCDSS)
- Monitoring Subsystem (MSS)
- Critical Event Handling Subsystem (CEHSS)
- Fault-Tolerant Decision Subsystem (FTDSS)

Both, the CCDSS and the CEHSS periodically produce output used to determine the behavior of the self-driving car. The MSS monitors the output of both, the CCDSS and the CEHSS. The FTDSS receives the output of the CCDSS and the CEHSS as well as the output of the monitoring of the MSS. In absence of failures, the CCDSS will provide its output to the FTDSS, the FTDSS will have this output checked by the MSS. In a failure-free scenario, the MSS will approve the CCDSS output and inform the FTDSS. The FTDSS, will then forward this MSS-approved output to the receivers (e.g., the actuators). A simple protocol in the receivers selects one output per cycle.

The CCDSS, the MSS, and the CEHSS form one fault-containment unit (FCU) each. This means, if some portion of a subsystem fails, then we consider the complete subsystem to be faulty. For example, if the CCDSS was realized as a stand-alone electronic control unit (ECU) and let this ECU implement a special purpose chip for object recognition, then in case this chip fails, the complete ECU is considered to be faulty.

A safety architecture must also define the failure behavior of FCUs. In our case, CCDSS, MSS, and CEHSS can fail-arbitrarily: such a faulty subsystem my send any sequence of messages on their interfaces to the FTDSS. The FTDSS itself is composed of two FCUs with limited failure behavior, which can be established by common fail-safe technologies like lock-step mechanisms.

Even in this rather simple architecture, there are many different failure scenarios. In some scenarios, for example, the CCDSS will fail to produce a safe output and the MSS will detect this failure. In these scenarios the FTDSS will forward the output of the CEHSS rather than the output of the CCDSS. As there are many different failure scenarios, the manual inspection of all of them is cumbersome and error-prone: some scenarios may easily be overlooked or interpreted incorrectly. Thus, we have used model-checking to exhaustively explore all possible failure scenarios. We have invented this approach almost twenty years ago in the context of network protocol verification in [2].

Using exhaustive fault simulation we have shown that a system implementing the Kopetz architecture is guaranteed to satisfy the following properties if only at most one FCU fails:

- consistency: all non-faulty receivers will use the same values and these values will be safe,
- termination: all non-faulty receivers will use some values at the end of each cycle, and
- non-triviality: all non-faulty receivers will use as values only output provided by the FTDSS FCUs.

It is noteworthy to add that the proposed architecture enables the implementation of a simple protocol in the receivers to pick one of potentially many outputs from the different FTDSS FCUs per cycle. It is simple in a sense that the receivers do not need to exchange information between themselves to achieve the consistency property as defined above. Typically, two rounds of communication are required, but the structure of the architecture simulates these two rounds by a first communication between the CCDSS, MSS, and CEHSS to the FTDSS FCUs and by a second communication from the FTDSS FCUs to the receivers. In that regard, the architecture shares similarities with the NASAS ROBUS architecture [3].

IV. Outlook - Standardization

Safety standards can have two roles. On the one hand they give guidance on how to build a safe system. On the other hand, they allow the comparison of different solutions with each other. A standardized safety architecture can define a minimum quality level to avoid safety shortcuts in particular. Of course, that will only be the case when the standard is indeed accepted and adopted by the automotive industry. As a response to these safety system requirements, the Kopetz architecture is proposed at The Autonomous, an open platform that aims at building an ecosystem of all players involved in the development of safe autonomous mobility.

References

[1] H. Kopetz, “An Architecture for Driving Automation,” https://www.the-autonomous.com/news/an-architecture-for-driving-automation/, accessed: 2021-12-13.

[2] W. Steiner, J. Rushby, M. Sorea, and H. Pfeifer, “Model checking a fault-tolerant startup algorithm: From design exploration to exhaustive fault simulation,” in *International Conference on Dependable Systems and Networks, 2004*. IEEE, 2004, pp. 189–198.

[3] P. S. Miner, M. Malekpour, and W. Torres, “A conceptual design for a reliable optical bus (robus),” in *Proceedings. The 21st Digital Avionics Systems Conference*, vol. 2. IEEE, 2002, pp. 13D3–13D3.