COMPUTER-AIDED DESIGN OF FAULT-TOLERANT HARDWARE ARCHITECTURES FOR AUTONOMOUS DRIVING SYSTEMS

Julitz, Tim Maurice;
Tordeux, Antoine;
Löwer, Manuel

University of Wuppertal

ABSTRACT

Fault-tolerant hardware architectures for autonomous vehicles can be implemented through redundancy, diversity, separation, self-diagnosis, and reconfiguration. These approaches can be coupled with majority redundancy through M-out-of-N independent system architectures. The development of fault-tolerant systems is of central importance in the launch of autonomous driving systems from level 4. The increasing complexity of electrical and electronic systems is challenging for the design of safety-critical systems. This work aims to develop a method to manage this complexity in product development and to use it to compare different types of architectures. The basis is a system consisting of sensors and microcontrollers. The reliability of all possible MooN configurations of the system is calculated automatically by numerically solving the master equation of the corresponding Markov chain. Subsequently, a software-based fault tree analysis enables more detailed modeling of the component structure. The results show that four-line architectures can provide suitable results and that the development effort for 2-ECU systems is higher than for 1-ECU systems with respect to the ISO 26262 target values.

Keywords: Autonomous driving, Fail-operational, Product architecture, Computational design methods, Numerical methods

Contact:
Julitz, Tim Maurice
University of Wuppertal
Germany
julitz@uni-wuppertal.de

Cite this article: Julitz, T. M., Tordeux, A., Löwer, M. (2023) 'Computer-Aided Design of Fault-Tolerant Hardware Architectures for Autonomous Driving Systems', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.105

1 INTRODUCTION

The degree of automation of road vehicles is divided into 5 SAE levels. With the elimination of the human fallback level in the SAE 4 level, the development of a fault-tolerant system architecture remains one of the key challenges for the market introduction of autonomous vehicles (Daily et al., 2017). Vehicle systems up to automation level 3 are currently on the market, in which a driver has to take over depending on the situation. Level 4 systems with no human fallback system are expected to be available in 2030 (Esser and Kurte, 2018). Fault tolerance can be implemented using a variety of strategies. One of them is the use of redundant hardware architectures that can detect failures in order to be able to keep running with the working hardware components. This requires multi-core CPUs, which are compared using majority redundancy (M-out-of-N). In a literature analysis, English language databases of microelectronics and automotive engineering were searched. A variety of approaches for fault-tolerant hardware architectures could be found. (Baleani et al., 2003) analyzes different architectures with regard to costs, performance, fault coverage and flexibility and distinguishes between systems that consist of one or more chips. (Ishigooka et al., 2018) focuses on reducing CPU load through cost-effective multi-mode architectures. (Kohn et al., 2015) presents a multi-stage majority decision process that also takes single and dual-chip systems into account. (Lin et al., 2018) accelerate hardware architectures with graphics processing units (GPUs), field programmable gate arrays (FGPAs) and application specific integrated circuits (ASICs), which can also be modeled as majority decision systems. When analyzing hardware architectures, (Schmid et al., 2019) uses fault trees to refer to the requirements of ISO 26262. (Sari, 2020) develops an approach for dependent failure analysis according to ISO 26262 using a hardware architecture at component level. The analysis of the literature showed that all of the examined architectures can be reduced to two types. Examples of each type are shown in the figs. 1 and 2. Fig. 1 represents a 2-out-of-2 (2oo2) dual-fail-safe architecture consisting of two independent subsystems, each consisting of an electronic control unit (ECU).

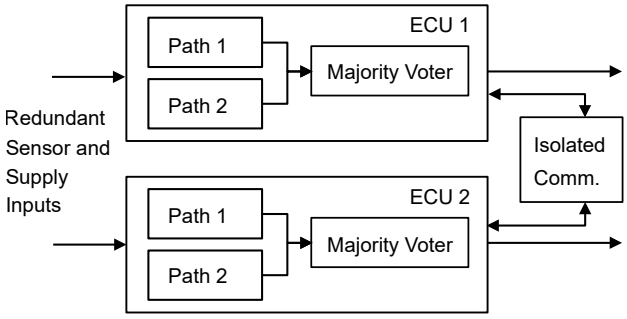

Figure 1. 2oo2 dual-fail-safe architecture (DFS) with isolated communication

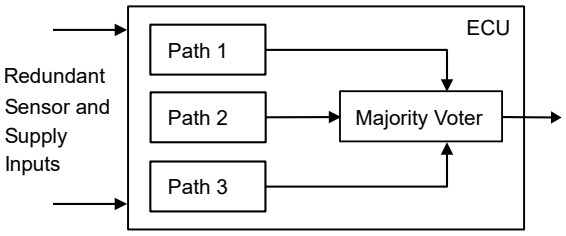

Figure 2. Tri modular 2oo3 architecture (TMR)

If both ECUs are working, the system runs in fail-operational mode. Redundant sensor inputs are handled by two independent ECUs that are independently powered. Each ECU processes the input in two redundant paths that are compared by a 2oo2 majority voter. Each path is represents a microcontroller unit (MCU). If an ECU is switched off due to a failed majority comparison, the system is in the fail-safe state. The advantage of multi-ECU systems is that they can be structurally or functionally separated. The second type of architecture (fig. 2) is based on a single ECU, which is often built in a trimodular redundancy (TMR). A single ECU is supplied by redundant sensor inputs. The majority comparison takes place on the same chip, but the possibility of functional diversity is limited. Reliability can be improved by any increase in the comparison threshold (MooN). In addition, this architecture is more cost-effective and space-saving. The presented ECU-architectures are supplied by sensor information. The established sensor architectures consisting of a combination of camera, radar and lidar are being replaced by new concepts. For example, Intel proposes a system that can operate independently with just a camera or just radar and lidar (Mobileye, 2022) while Tesla vehicles only rely on camera vision (Tesla, 2022). Accidents with comparable vehicle systems indicate a need for research (National Transportation Safety Board (NTSB), 2017). The aim of this work is to investigate the following research questions:

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

1. Which approach is suitable for systematically and automatically analyzing and modeling system architectures with MooN redundancy?
2. What configuration of MooN redundancy of sensors and microcontroller units (MCUs) is required to ensure fail-safe operation for driverless applications?
3. Which ECU architecture (TMR or DFS) with which number of sensors and MCUs is best suited to meet the requirements of Automotive Safety Integrity Level D (ASIL D) of ISO 26262?

Question i represents a core question of the article. The complexity in modern vehicle engineering is increasing exponentially (Laissy et al., 2022). But not only the automotive industry is affected. In general, product development also faces major challenges in view of the increasing complexity (Trattner et al., 2019). This work provides a contribution to complexity control in product hardware development with a new method using the example of an autonomous vehicle system.

2 METHOD

2.1 General approach

The developed method represents a top-to-bottom approach, which is demonstrated using electronic control units (ECUs) for vehicle systems. Since comparable microelectronics in the form of microcontrollers (MCUs) play a major role in modern products, the method will be generally applicable (Zhang and Wu, 2021). First, the identified hardware architectures are evaluated by Markov chains to roughly design the system structure (sec. 2.2). The Markov process is systematically modeled with an R script. Markov chains are a mathematical model for analyzing complex systems that change over time. In the context of analyzing the reliability of M out of N majority redundant hardware architectures, Markov chains provide a powerful tool for computing probabilities based on the current state of the system. The relevance of Markov chains for reliability analysis lies in the stochastic nature of these systems, where the failure of one component can trigger a cascade of failures, ultimately leading to the failure of the entire system. Markov chains can model these complex dependencies and provide an accurate assessment of system reliability.

After the rough system architecture is in place, the component structure is modeled in detail using computer-aided fault tree analysis (sec. 2.3). Fault tree analysis (FTA) is an established graphical tool to identify and analyze the causes of system failures. Related to the design of a detailed component structure for Electronic Control Units (ECUs), FTA can help identify potential failure modes, their causes and their effects, which is critical to ensure the safety of complex systems. The FTA model should be verified to meet the requirements of relevant standards such as ISO 26262 that requires the use of FTA to identify potential hazards and risks in automotive systems. Compared to alternative approaches, FTA offers a more comprehensive and reliable approach to designing the component structure of ECUs. Alternative approaches based on experience or using ad hoc methods are less suitable for several reasons:

1. Limited scope: Experience-based methods may not consider all potential failure modes, which result in critical failure modes being overlooked.
2. Lack of structure: Ad hoc methods lack a structured approach to failure mode identification and analysis, which can lead to inconsistencies in analysis and design.
3. Subjectivity: Methods based on experience and intuition are subjective and rely on the knowledge and prejudices of the individual. This can result in an unreliable analysis and design that may not account for all potential failure modes.
4. Lack of standardization: Ad hoc methods lack a standardized approach, which can lead to inconsistencies in analysis and design and make it difficult to compare the results of different designs or projects.

Overall, the combination of computer-aided markov and fault tree analysis provides a more comprehensive and structured approach to modeling hardware architectures in terms of safety, reliability and complexity management.

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

2.2 Evaluation of hardware architectures with Markov chains

Various hardware architectures are analyzed using Markov chains. For this purpose, the architectures are divided into two parts modeled by majority redundancy. Sensor architecture redundancy is reffered to below as $S\text{oo}N_S$ while the MCU architecture redundancy is reffered to as $M\text{oo}N_M$. Here N_S and N_M are the total number of sensors or MCUs; S and M are the number of sensors and MCUs required for system operation. This approach does not take into account the type of sensors such as radar, lidar or camera. It is aimed precisely at the number of sensors that are required for fault-tolerant operation of the driving system. The same applies to the number of MCUs. In addition, the Markovian model assumes that the MCU and sensor failure rates are constant over time. More realistic models could include time or state dependent rates. With the help of Markov processes, the number of sensors and MCUs and their redundant configuration in combination is varied to evaluate the reliability of the overall system. For example, a system consisting of 3 sensors and 4 MCUs, which is operational when at least 2 sensors and 3 MCUs are working, is modeled as a 2oo3/3oo4 system. The term $S\text{oo}N_S/M\text{oo}N_M$ is used below to describe the full systems. Systems with N_M identical MCUs and N_S identical sensors are condidered. The time to failure of the MCUs and sensors are distributed identically exponentially with $\lambda_M > 0$ as the failure rate of a MCU and $\lambda_S > 0$ as the failure rate of a sensor. In the following, $P_{m,s}(t)$ is the state probability of $m = 0,\ldots N_M$ MCU(s) and $s = 0,\ldots,N_S$ sensors are working at time t. The sum of all $P_{m,s}(t)$ is equal to 1 for all t, i.e.,

$$\sum_{m=1}^{N_M}\sum_{s=1}^{N_S} P_{m,s}(t) = 1. \tag{1}$$

The definition of a system failure depends on the majority redundancy used. The reliability of a $S\text{oo}N_S/M\text{oo}N_M$ system is described by Eq. 2.

$$R(t) = \sum_{m=M}^{N_M}\sum_{s=S}^{N_S} P_{m,s}(t). \tag{2}$$

The system contains up to $(N_M + 1)(N_S + 1)$ possible states. The transitions from one state to another are determined according to a transition rate matrix. For example, the transition rate matrix A for a system with 2 sensors and 3 MCUs is shown in Eq. 3 according to the state transition diagram in Fig. 3 of the system in Fig. 4.

$$A = \tag{3}$$

	S2/M3	S2/M2	S2/M1	S2/M0	S1/M3	S1/M2	S1/M1	S1/M0	S0/M3	S0/M2	S0/M1	S0/M0
S2/M3	$-2\lambda_S - 3\lambda_M$	$3\lambda_M$	0	0	$2\lambda_S$	0	0	0	0	0	0	0
S2/M2	0	$-2\lambda_S - 2\lambda_M$	$2\lambda_M$	0	0	$2\lambda_S$	0	0	0	0	0	0
S2/M1	0	0	$-2\lambda_S - \lambda_M$	λ_M	0	0	$2\lambda_S$	0	0	0	0	0
S2/M0	0	0	0	$-2\lambda_S$	0	0	0	$2\lambda_S$	0	0	0	0
S1/M3	0	0	0	0	$-\lambda_S - 3\lambda_M$	$3\lambda_M$	0	0	λ_S	0	0	0
S1/M2	0	0	0	0	0	$-\lambda_S - 2\lambda_M$	$2\lambda_M$	0	0	λ_S	0	0
S1/M1	0	0	0	0	0	0	$-\lambda_S - \lambda_M$	λ_M	0	0	λ_S	0
S1/M0	0	0	0	0	0	0	0	$-\lambda_S$	0	0	0	λ_S
S0/M3	0	0	0	0	0	0	0	0	$-3\lambda_M$	$3\lambda_M$	0	0
S0/M2	0	0	0	0	0	0	0	0	0	$-2\lambda_M$	$2\lambda_M$	0
S0/M1	0	0	0	0	0	0	0	0	0	0	$-\lambda_M$	λ_M
S0/M0	0	0	0	0	0	0	0	0	0	0	0	0

A component failure reduces the count by one. Only one component can fail at a time. For example, a sensor failure from the initial state leads to the transition S2/M3 → S1/M3. A failed component cannot be recovered. Therefore only transitions $(m,s) \mapsto (m-1,s)$, $m > 0$, and $(m,s) \mapsto (m,s-1)$, $s > 0$, are possible. Consequently, the Markov process transition matrix A associated with the systems is triangular and mostly sparse. The vector of the probabilities of the different possible states is given by

$$P = (P_{N_M,N_S}, P_{N_M-1,N_S}, \ldots, P_{0,N_S}, P_{N_M,N_S-1}, \ldots, P_{0,0})^{\mathrm{T}}. \tag{4}$$

The state probabilities $P(t)$ are the solution of the Kolmogorov backward-linear differential equation (master equation)

$$\dot{P}(t) = A^{\mathrm{T}}P(t), \qquad P(0) = P_0. \tag{5}$$

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

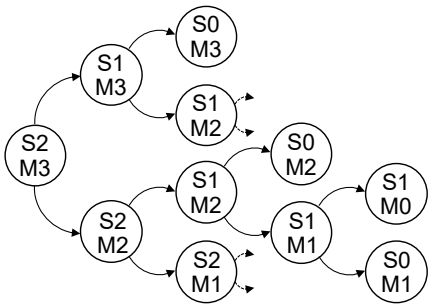

Figure 3. State transition diagram of a Markov chain for a system with 2 sensors and 3 MCUs.

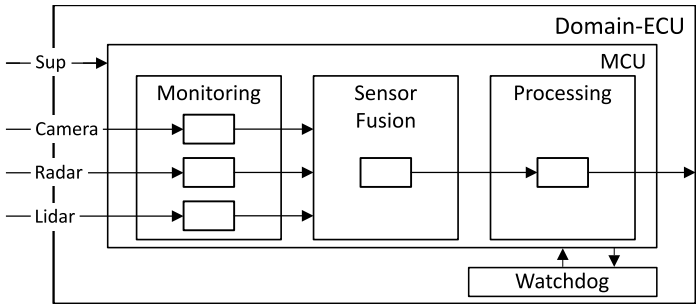

Figure 4. Components of a microcontroller (MCU) of an electronic control unit (ECU) of a fault-tolerant system architecture. Based on (Sari, 2020).

Here P_0 is the initial state of the system. We obtain directly the solution

$$P(t) = e^{A^{\mathrm{T}}t} P_0. \tag{6}$$

The calculation of the matrix exponent can be performed iteratively since the transition matrix is triangular. More generally, the matrix exponential can be obtained from Eq. 6 using linear algebra and matrix diagonalization, Laplace transform, or numerically on computer software. Note that the exponential form of the solution comes from the linear (Markovian) differential structure with constant failure rates. More realistic models could include time or state dependent rates. In such cases, except for certain models, the master equation must be calculated numerically. Finally, the global system reliability is determined with the state probabilities Eq. (6) and Eq. (2). We address the reliability of the sensor and MCU architecture by varying the number of MCUs with failure rate λ_M and the number of sensors with failure rate λ_S and their majority redundancy configuration. The resolution of the Markov process and the calculation of the system reliability according to Eq. (2) is done numerically A generic script given in listing 1 (f.) allows to set up the number of Sensors N_S, the number of MCUs N_M and their respective failure rates λ_S and λ_M. With this given input, the script generates the reliability curves $R(t)$ of all possible variations of the number of hardware components and their redundancy.

2.3 Fault-tree design of the hardware part structure

After the configuration of the system has been evaluated with the number of its components, the results can be used in the control unit architectures mentioned above (Fig. 1 and 2). Figures 1 and 2 show a dual-fail-safe architecture and a tri-modular system-on-a-chip architecture with variable sensor numbers and path numbers representing MCUs. With the findings of the Markovian analysis, a suitable configuration of both control unit architectures is modeled with fault trees. The aim is to meet the ASIL D reliability hardware requirements of ISO 26262-5 (2018) with the given amount of hardware components to compare both architectures in terms of reliability, the number of components needed to meet the ASIL D requirement are needed. The investigations are also based on the possible use of the architecture for fault-tolerant applications, e.g. Separation, diversification or self-diagnosis. In order to take the ISO 26262 target values into account, the component structure of the architectures from Fig. 1 and 2 is modeled at the component level. To do this, the paths, each of which represents an MCU, are further refined into their components. Fig. 4 shows the components of a single MCU based on the suggestions of (Sari, 2020).

Each MCU is powered by an independent power supply (Sup) supply and by the sensor information. The functionality is checked by a watchdog. First, the sensor information is monitored to be linked by a sensor fusion module. In the processing block, the execution of the control unit-specific function, which is kept solution-neutral here, is calculated. The MCU then forwards the processed sensor information to the majority decision. How many MCUs are modeled and which redundant configuration is used depends on the results of the preliminary Markov analysis. The fault tree analysis is performed by a framework in R (OpenReliability.org, 2022). First, a failure rate of $10^{-8}\ h^{-1}$ is set for each component according to the element-level target of ASIL D. A cut-set analysis is carried out in which it is systematically determined how many components must fail for the entire system to fail. The fault tree analysis is ideally suited for this. It will focus on single and two-point failures, meaning one or

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

Listing 1 R-script of a system with number of sensors `nbS`, failure rate of sensors `lambdaS`, number of MCUs `nbM`, and failure rate of MCUs `lambdaM`.

```
library(Matrix)

## Setting of the parameters
lambdaS=1e-8;lambdaM=1e-8
nbS=2;nbM=3;n=(nbS+1)*(nbM+1)

## Construction of the transition matrix A
A=matrix(0,n,n)
A[col(A)==row(A)+1]=c(rep(nbM:0*lambdaM,nbS),nbM:1*lambdaM)
A[col(A)==row(A)+nbM+1]=floor((n-1:(n-nbM-1))/(nbM+1))*lambdaS
A[col(A)==row(A)]=-apply(A,1,sum)

## State vector P(t)=exp(t(At))P0 for the initial state P0=(1,0,...,0)
P=function(t)
        expm(t(A*t))[,1]

## Reliability function
R=function(t,s,m)
        sum(P(t)[nbM-m>=(0:(n-1))%%(nbM+1)&(1:n)<=(nbS-s+1)*(nbM+1)])
R=Vectorize(R)

## Plotting of the reliability functions
le="1oo1/1oo1";co=1
curve(exp(-(lambdaS+lambdaM)*x),xlim=c(0,4e8),ylim=c(0,1),xlab="t",ylab="R(t)")
for(s in 1:nbS)
        for(m in 1:nbM){
                co=co+1
                curve(R(x,s,m),type='b',col=co,lty=co,pch=co,add=T)
                le=c(le,paste(s,"oo",nbS,"/",m,"oo",nbM,sep=""))}
legend("topright",le,lty=1:co,col=1:co,pch=c(-1,2:co))
```

two component failures will result in a system failure. This results in a grouping of the components according to their importance for the system. Single and two-point failure components are referred to below as first and second order components. The reliability of the first and second order components must be assigned according to the ISO 26262 target values $\lambda = 10^{-10}\ h^{-1}$ and $\lambda = 10^{-8}\ h^{-1}$. Based on the cut-set analysis, the architectures can be compared in terms of their suitability and effort to achieve these requirements.

3 RESULTS

3.1 Markov analysis of the architectures

The analysis of the architectures with the Markov chains is performed with a system consisting of a number of sensors up to N_S and a number of MCUs up to N_M. The variants of the resulting majority redundancies are systematically calculated by the R script. Fig. 5 is using the notation $SooN_S/MooN_M$ for the combination of sensor and MCU redundancy. The maximum number of sensors and MCUs is limited to 4 to account for cost and reduce complexity. The initial state represents a system for which all sensors and MCUs are working. The failure rate for sensors and MCUs is set to $\lambda = 10^{-8}\ h^{-1}$ due to the ISO 26262 target values. Changing these values would have no qualitative impact. Only the scaling of the timeline would change. Choosing different rates for sensors and MCUs would entail a change that was not made here due to the comparability of the architectures. In the next step of the fault tree analysis, the failure rates are varied. A reference is defined to have a comparison for the analysis of the architectures. It consists of a 1oo1/1oo1 system (black curve), i.e. a sensor and an MCU without redundancy. This reference can be used to assess whether the reliability has increased or decreased as a result of the architectures. The reliability curves of the architectures below the reference are not considered further. These are systems with a large serial part or an exclusively serial part ($NooN$) that are known to have a high probability of failure. On the opposite side are the curves with the highest reliability, which also cannot be taken into account. These are parallel systems without majority voting ($1ooN$). An error diagnosis cannot be carried out here. Consequently, the fault-tolerant suitability is missing. The focus of the analysis is on the $MooN$ architectures with $1 < M < N$. The architectures remaining after this limitation are 2oo3/2oo3, 2oo3/2oo4, and 2oo3/3oo4. All other configurations are

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

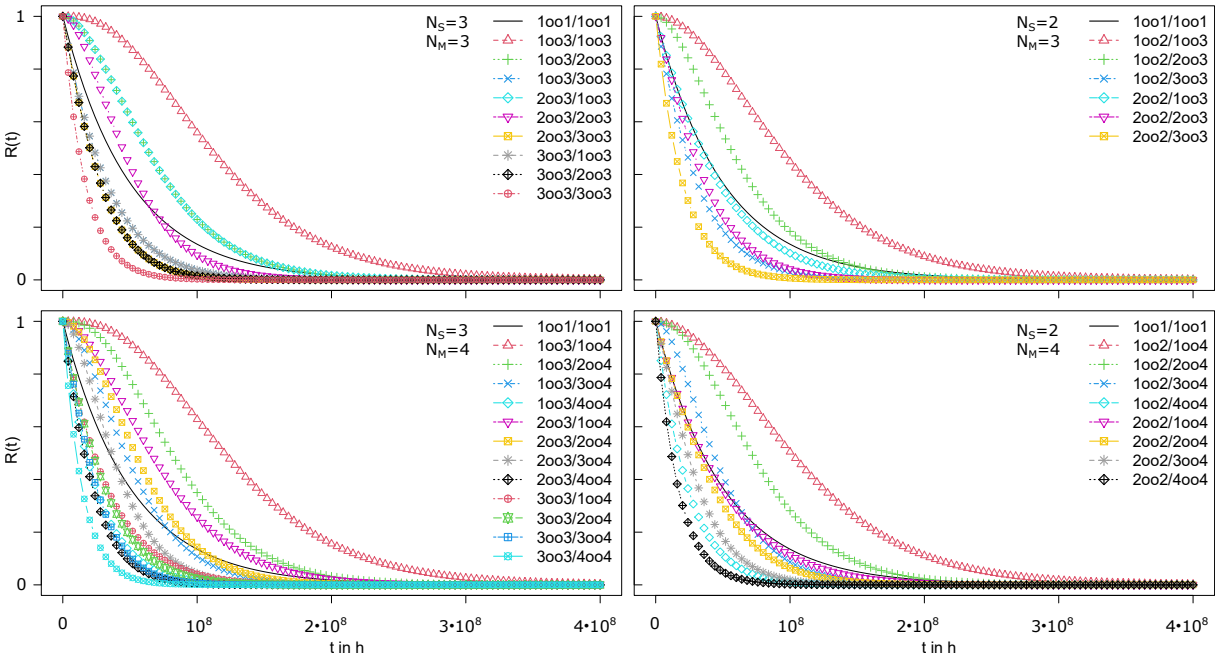

Figure 5. Survival probability $R(t)$ of majority redundant system architectures with the notation $SooN_S/MooN_M$ for Sensor/MCU redundancy. Number of sensors: up to N_S. Number of MCUs: up to N_M with $\lambda_S = 10^{-8}\ h^{-1}$ and $\lambda_M = 10^{-8}\ h^{-1}$.

pure series or parallel systems. It can be seen that this does not involve an architecture with only two sensors. This is due to the limitation that a majority decision is required. In practice, this problem can be solved by a dual-fail-safe architecture using 2 x 2 sensors, resulting in an overall higher hardware usage. Another way to compare architectures is through mathematical expectation. For an exponential distribution it is defined by $E(X) = 1/\lambda$. For the given failure rate it is $t = 10^8$ h. It can now be observed which curves are above the reference at this point in time. Most of the curves are above the reference for the system with 3 sensors and 4 MCU. The 2oo3/2oo4 is again confirmed as suitable. The other two suitable architectures previously identified are below the reference. It is striking that the 1oo2/2oo4 is very reliable in comparison. A solution to make this architecture mostly comparable can in turn be a dual-fail-safe architecture.

3.2 Fault tree analysis of the hardware parts

The following fault tree analysis focuses on the identified 2oo3/2oo4 architecture and can be applied to all other architectures in the same way. According to Fig. 1 and 2, the 2oo3/2oo4 is modeled as a dual-fail-safe and a single-ECU in TMR style. The individual control unit is modeled directly as 2oo3/2oo4 and referred to below as 1-ECU. The dual-fail-safe architecture is modeled as a double 2oo3/2oo2 architecture (see Fig. 1) and referred to as 2-ECU in the following. The individual ECUs are set up as shown in Fig. 4. A section of the 2-ECU fault tree is shown as an example in Fig. 6. Due to their complexity, the fault trees cannot be fully represented. The elementary events are marked by circles. In the excerpt shown, this is only the isolated communication, which has a failure rate of $10^{-10}\ 1/h$ because it is a one-point failure. The blue fields represent hidden branches of the fault tree. The green notation “prob" is the survival probability calculated by the R framework. In a later step, the survival probability of the top level is calculated back to the system failure rate. The script-based modeling allows the calculation of the fault tree over time using for loops. First, the cut sets of the 1-ECU and the 2-ECU systems are determined. The 1-ECU system contains two first-order cut sets. They consist of the processor core of the second voting stage and the power supply of the voting MCU. Four cut sets make up the second order and the third order failures consist of 112 cut sets. For the 2-ECU system there is a first-order cut set, the isolated communication. The number of second-order cut-sets is 67 and the number of third-order cut-sets is 144. It can be seen that the 2-ECU system has significantly more second-order cut sets than the 1-ECU system, which must be designed to be correspondingly more

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

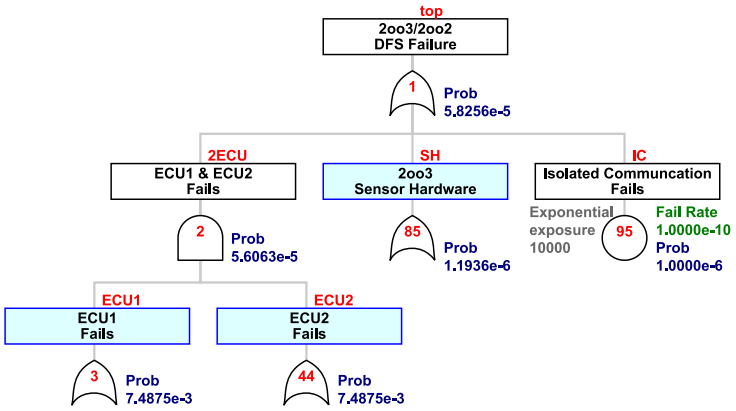

Figure 6. Top-level sector of a 2oo3-Sensor and 2oo2-Dual ECU fault tree

reliable in order to achieve the same result. The failure rates of the components are approximated on this basis. Three groups are defined: Cut Set Order 1 components (single point failures), Cut Set Order 2 components (two point failures) and all other components. Each group is assigned a single failure rate. Order 1: $10^{-10}\ h^{-1}$, Order 2: $10^{-7}\ h^{-1}$, Order 3: $10^{-8}\ h^{-1}$ according to ISO 26262. Fig. 7 (left panel) shows the result of the fault tree analysis for the 1-ECU system. The blue solid curve shows the system

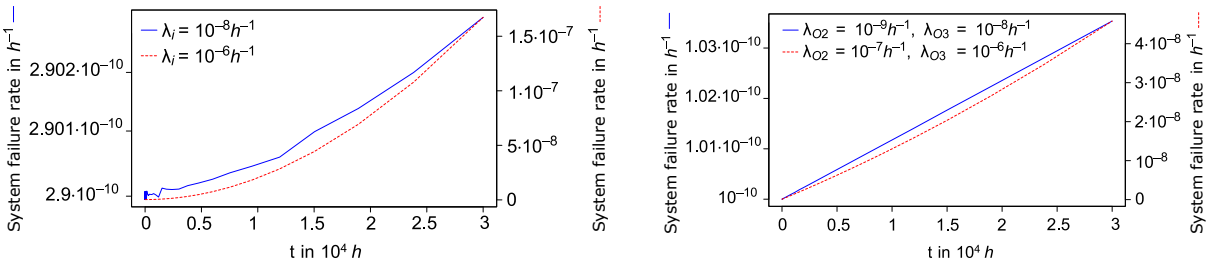

Figure 7. Development of the system failure rate over time for the 2oo3/2oo4 1-ECU system using two different failure rates (left panel) and for the 2oo3/2oo2 DFS 2-ECU system using four different failure rates and components of the second and third cut-set order (right panel).

failure rate of the 1-ECU system over time using a component failure rate of $10^{-8}\ h^{-1}$. There are no single point failures. The failure rate for two-point failures is neglected since only two components are affected. The system failure rate at the top level of the fault tree is about $3 \cdot 10^{-10}\ h^{-1}$. The component failure rate can thus be reduced in accordance with the target value of ISO 26262. With the rate reduced to $10^{-6}\ h^{-1}$ (red dashed line), the system failure rate is approaching the targeted $10^{-8}\ h^{-1}$. It can be stated that in the present 1-ECU system the failure rate of the components of the third and higher cut-set order can be reduced to $10^{-8}\ h^{-1}$ in order to achieve the ASIL D target value. Fig. 7 (right panel) shows the result of the fault tree analysis for the 2-ECU system. Since there are significantly more cut sets of the second order (67) than in the 1-ECU system, a distinction is made between components of cut set order two and three for the uniform failure rate. Second order components are designed with a lower failure rate than third order components. First, the target value of $10^{-8}\ h^{-1}$ is used again (blue, solid curve). The third order components are accordingly provided with a failure rate of $10^{-9}\ h^{-1}$. Here, too, the system failure rate levels off at around $10^{-10}\ h^{-1}$. To approximate the target system failure rate of $10^{-8}\ h^{-1}$, the third-order cut-set component failure rates can be set to $10^{-6}\ h^{-1}$ and the second-order cut-set component failure rates are reduced to $10^{-7}\ h^{-1}$ (red, dashed curve).

4 DISCUSSION

As part of the Markov analysis, it was found that the reliability of redundant architectures is grouped according to the proportion of serial and parallel structures. Systems with a high serial content (1oo*N*)

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

have a low probability of survival, while systems with a high parallel content (Noo$N$) have a high probability of survival. However, for fail-operational applications, a majority decision is required, which is why the choice of the Moo$N$ architecture is subject to the following restrictions: $1 < M < N$. It has been found that 2oo3/2oo3, 2oo3/2oo4 and 2oo3/3oo4 have the greatest reliability. Also of note is that Moo$N$ systems perform significantly better with a N of 4 than with a N of 3. The fault tree analysis found that the 2-ECU architecture performs significantly more two-point failures than the 1-ECU architecture. For an ISO 26262 compliant design, the 2-ECU architecture requires more effort and cost. With the method presented, a wide variety of architectures can be analyzed and modeled. Since the methodology is based on a component-independent analysis of architectures, it can be used universally. In the second step, an individual component structure can be created for the respective application. The findings are limited to components with identical failure rates. However, the methods presented are robust to heterogeneous systems. Further restrictions lie in the general restriction of redundant configurations due to dependency effects of the subsystems. Such a phenomenon was recently empirically proven for the multi-sensor perception of autonomous vehicles (Gottschalk et al., 2022). The fault tree representation at different levels enables a new approach for assigning the target values of ISO 26262 for single and two-point failures. In addition to identifying one and two-point failures with cut sets, the levels of the fault tree can also be viewed individually. Entire groups of components can thus be viewed as a one- or two-point failure. The failure rate target can then be assigned as a group. If you look at Fig. 6, you can see that in addition to the single-point failure of the components of the isolated communication, the component groups ECU1 and ECU2 and the entire sensor hardware can also be affected and assigned as individual failures. This approach offers new possibilities when applying the ASIL decomposition according to ISO 26262.

5 CONCLUSION

The developed method offers a systematic and comprehensive approach to modeling hardware architectures with regard to safety, reliability and complexity management. The technique is demonstrated using electronic control units (ECUs) for vehicle systems, but can also be applied to other products that use comparable microelectronics in the form of microcontrollers (MCUs). The combination of computer-aided Markov and fault tree analysis offers several advantages over alternative approaches. Markov chains provide a powerful tool for computing probabilities based on the current state of the system, which can model complex dependencies and provide an accurate assessment of system reliability. Fault tree analysis, on the other hand, provides a structured approach to identifying and analyzing failure modes that can help ensure potential hazards and risks are identified and addressed. The method is particularly relevant for the design of the component structure of ECUs to ensure fail-safe operation for driverless applications. The part of the approach regarding the fault tree analysis could only partially fulfill the research question. It allows systematic analysis and modeling of the architectures at the component level, but this is done manually. Since the fault trees are also based on an R script, automation is basically possible and is planned for future projects. The script-based approach also allows the consideration of different failure rate models, such as the Weibull distribution, where the transition rate matrix is time dependent. This enables a more realistic modeling of the systems. Since the system is no longer linear in the case of a time-dependent transition matrix, its resolution requires the use of numerical approximations for estimating the state probabilities. With the help of the fault tree analysis, it was possible to determine that both the 1-ECU and the 2-ECU architecture are suitable for fault-tolerant operation from a reliability point of view. However, the 2-ECU requires a higher use of resources because more components must have a lower failure rate due to the higher number of two-point failures.

REFERENCES

Baleani, M., Ferrari, A., Mangeruca, L., Sangiovanni-Vincentelli, A., Peri, M. and Pezzini, S. (2003), "Fault-tolerant platforms for automotive safety-critical applications", in: J. Moreno, P. Murthy, T. Conte and P. Faraboschi (Editors), *Proceedings of the international conference on Compilers, architectures and synthesis for embedded systems - CASES '03*, ACM Press, New York, New York, USA, p. 170, http://doi.org/10.1145/951710.951734.

Daily, M., Medasani, S., Behringer, R. and Trivedi, M. (2017), "Self-driving cars", *Computer*, Vol. 50 No. 12, pp. 18–23, http://doi.org/10.1109/MC.2017.4451204.

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press

Esser, K. and Kurte, J. (2018), "Autonomous driving. current status, potentials and impact analysis. study for the association of german chambers of industry and commerce e.v. (translated)", [online]. https://www.dihk.de/resource/blob/3924/b1d16ab3418ee25133fe2efdfa04c832/studie-autonomes-fahren-data.pdf

Gottschalk, H., Rottmann, M. and Saltagic, M. (2022), "Does redundancy in AI perception systems help to test for super-human automated driving performance?", *Deep Neural Networks and Data for Automated Driving*, pp. 81–106.

Ishigooka, T., Honda, S. and Takada, H. (2018), "Cost-effective redundancy approach for fail-operational autonomous driving system", in: *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, IEEE, pp. 107–115, http://doi.org/10.1109/ISORC.2018.00023.

ISO 26262-5 (2018), "Road vehicles—functional safety—part 5: Product development at the hardware level", International Standard.

Kohn, A., Kabmeyer, M., Schneider, R., Roger, A., Stellwag, C. and Herkersdorf, A. (2015), "Fail-operational in safety-related automotive multi-core systems", in: *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*, IEEE, pp. 1–4, http://doi.org/10.1109/SIES.2015.7185051.

Laissy, J.C., Lyon, V., Le Mouëllic, M., Walus, S., Alt, L. and Giraud, R. (2022), "How to deal with exponential complexity in automotive engineering", *LinkedIn Pulse*. https://www.linkedin.com/pulse/how-deal-exponential-complexity-automotive-jean-christophe-laissy-

Lin, S.C., Zhang, Y., Hsu, C.H., Skach, M., Haque, M.E., Tang, L. and Mars, J. (2018), "The architectural implications of autonomous driving", in: X. Shen, J. Tuck, R. Bianchini and V. Sarkar (Editors), *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, New York, NY, USA, pp. 751–766, http://doi.org/10.1145/3173162.3173191.

Mobileye (2022), "True redundancy: The realistic path to deploying avs at scale", [online], accessed 2022-11-10. https://www.mobileye.com/technology/true-redundancy/

National Transportation Safety Board (NTSB) (2017), "Highway accident report: Collision between a car operating with automated vehicle control systems and a tractor-semitrailer truck near williston, florida, may 7, 2016.", *NTSB/HAR-17/02*.

OpenReliability.org (2022), "Fault tree analysis on r", [online], accessed 2022-11-10. http://www.openreliability.org/fault-tree-analysis-on-r/

Sari, B. (2020), *Fail-operational Safety Architecture for ADAS/AD Systems and a Model-driven Approach for Dependent Failure Analysis*, Springer Fachmedien Wiesbaden, Wiesbaden, http://doi.org/10.1007/978-3-658-29422-9.

Schmid, T., Schraufstetter, S., Wagner, S. and Hellhake, D. (2019), "A safety argumentation for fail-operational automotive systems in compliance with iso 26262", in: *2019 4th International Conference on System Reliability and Safety (ICSRS)*, IEEE, pp. 484–493, http://doi.org/10.1109/ICSRS48664.2019.8987656.

Tesla (2022), "Transitioning to tesla vision", [online], accessed 2022-11-10. https://www.tesla.com/support/transitioning-tesla-vision

Trattner, A., Hvam, L., Forza, C. and Herbert-Hansen, Z.N.L. (2019), "Product complexity and operational performance: A systematic literature review", *CIRP Journal of Manufacturing Science and Technology*, Vol. 25, pp. 69–83, http://doi.org/10.1016/j.cirpj.2019.02.001.

Zhang, L. and Wu, H. (2021), "Application of single chip technology in internet of things electronic products", *Journal of Intelligent & Fuzzy Systems*, Vol. 40 No. 2, pp. 3223–3233, http://doi.org/10.3233/JIFS-189362.

https://doi.org/10.1017/pds.2023.105 Published online by Cambridge University Press