2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops

Safety engineering for autonomous vehicles

Rasmus Adler, Patrik Feth, Daniel Schneider
Embedded System Quality Assurance
Fraunhofer Institute for Experimental Software Engineering IESE, 67663 Kaiserslautern, Germany
rasmus.adler|patrik.feth|daniel.schneider@iese.fhg.de

***Abstract*— In safety engineering for non-autonomous vehicles, it is generally assumed that safety is achieved if the vehicle appropriately follows certain control commands from humans such as steering or acceleration commands. This fundamental assumption becomes problematic if we consider autonomous vehicles that decide on their own which behavior is most reasonable in which situation. Safety criticality extends to the decision-making process and the related perception of the environment. These, however, are so complex that they require the application of concepts for intelligence that do not harmonize with traditional safety engineering. In this paper, we investigate these problems and propose a solution.**

Keywords—safety engineering, autonomous driving, safety cage

I. Introduction

The number of advanced driver assistance systems, such as adaptive cruise control or lane assistance, is constantly increasing. If we take all such systems that can be found in a high-end car together, we are actually pretty close already today to fully automated driving (in certain circumstances such as driving on the highway). From here the next natural step appears to be the move from such a high degrees of automation to completely autonomous driving. If we look at other related domains, such as aviation or rail, it seems that we are even closer to this vision because the perception of the relevant environmental situations is generally easier and already much more established.

One important difference between human assistance and full automation is the assurance of safety with respect to the occurrence of harm, property damage, or other unwanted events. In the case of human assistance, driving or flying safely is the operator's responsibility. She monitors the current situation at any point in time and decides which behavior is safe in which situation. In case of full automation, the vehicle decides whether a certain behavior is safe in the current situation. The responsibility for making safe decisions is thus shifted from the operator to the manufacturer of the vehicle.

Thus, manufacturers of safety-critical autonomous vehicles need to know what to do in order to satisfy the additional responsibility that comes with the autonomous decision-making of their vehicles. Considering the behavior of non-autonomous vehicles, safety standards tell manufacturers what to do in order to satisfy their responsibility. Which safety standard has to be considered by a manufacturer depends on the type of vehicle being developed. Road vehicles have to be developed according to a different safety standard than agricultural machinery. However, independent of the concrete type of vehicle, the focus is always on malfunctions of some predictable behavior. This means that it is known at design time how the vehicle shall behave at runtime. Non-autonomous vehicles are generally designed to follow the control commands of the operator. As autonomous vehicles additionally assume the role of the operator, it is much more difficult and complex to understand and specify their (safe) behavior in any situation. This problem might even be aggravated due to the utilization of artificial intelligence technologies such as neuronal networks. In this case, it is even unpredictable at design time how the vehicle will behave in a certain runtime situation.

The transition from predictable to unpredictable behavior is thus not considered in safety standards and constitutes a problem for the assurance of safety. Current safety engineering and safety standards focus on *malfunctioning behavior*. However, the term *behavior malfunction* gets a totally different meaning if we transition from predictable to unpredictable behavior. Regarding predictable behavior, *behavior malfunction* is a deviation from some predefined (specified) behavior. Regarding unpredictable behavior, *behavior malfunction* must be interpreted differently because unpredictable behavior emerges at runtime and is, by definition, not specified explicitly at design time. As it depends on the current situation whether a certain behavior is desired or not, we consider *behavior malfunction* of unpredictable behavior as an unwanted combination of a certain behavior and a certain runtime situation. Due to the large number of possible behaviors and the huge number of possible situations, it is hardly possible to specify explicitly every unwanted combination of behavior and situation. Nevertheless, we propose identifying and specifying the most critical combinations explicitly in order to safeguard them with traditional means of safety engineering. Minor critical malfunctions have to be safeguarded by novel/future safety engineering means such as validation of neural networks.

In this paper, we first present an approach for identifying and specifying the most critical malfunctions. To this end, we handle the huge number of possible situations with a formal specification technique. Furthermore, we present an approach for implementing a *safety supervisor,* which detects these malfunctions at runtime. The approach handles failures of the complex recognition of the current situation so that omission failures of the malfunction detection are avoided and commission failures are minimized.

In Section 2, we justify our strategy to safeguard intelligent functions with a safety supervisor. In Section 3, we present our approach for identifying the most critical malfunctions and deriving a requirements specification for our safety supervisor.

In Section 4, we provide means for implementing the safety supervisor that detects the specified malfunctions according to their criticality. In Section 5, we present related work before summarizing and concluding in Section 6.

II. Approach Justification

Our fundamental strategy complies with traditional safety engineering, where the primary objective is not to prevent malfunctions, but to ensure that safety mechanisms are provided that identify faults and provide countermeasures in order to prevent damage to people and to the environment. For example, a suitable safety measure for a robotic arm that is capable of harmful movements is a physical safety cage that limits the range of motion of the robot to its work area. Such a safety cage can be implemented as a light curtain that invokes an emergency stop once it is touched. The focus of the safety-related development can then be limited to the safety mechanism itself. According to this approach, the domain-independent safety standard IEC 61508 focuses on the identification, specification, and correct implementation of dedicated safety functions.

However, the benefit of dedicated safety functions or mechanisms decreases with the complexity of the safety functions because the effort required to engineer the functions increases greatly and failures of the safety functions become more likely. Considering complex vehicle functions, such as the electronic stability control (ESP), the primary objective is therefore to prevent malfunctions of the normal functions and not to realize simple safety functions that would safeguard against all possible failures of the normal functions. Accordingly, the functional safety standard ISO 26262, which is the automotive domain-specific derivation of IEC 61508, focuses on safety requirements and their assignment to functions. As these functions are not necessarily simple safety functions, they are often referred to as safety-related functions.

Simple safety functions nevertheless play an important role in the safety assurance of modern vehicles. One reason for this is that they can safeguard against the most critical failures of a normal function and thus lower the criticality of the normal function. For instance, the most critical failure of the ESP is that the differences between the moments of the wheels exceed a certain threshold. A simple safety function that limits the difference of the wheel moments can avoid this most critical failure and thus lower the criticality of the ESP. The reduction of the criticality of a normal function by means of a safety function is also considered in ISO 26262. As we will explain in the next section, the normative concept is the ASIL decomposition [1].

We assume that simple safety functions will also play an important role in the safety assurance of intelligent functions that create unpredictable behavior. One reason for this is that we expect to be able to identify and safeguard the most critical malfunctions of unpredictable behavior and thus lower the criticality of the intelligent functions generating this behavior. This application of safety functions corresponds to the one described above. Another reason for using simple safety functions is the lack of techniques for creating confidence in unpredictable behavior. The main technique is validation, as verification requires an explicit specification. Due to the huge number of situations, the emergent behavior can usually not be evaluated in all situations. A safety argumentation that is based purely on validation might thus be hard to advocate and would be supported better by simple safety functions that safeguard against the most critical malfunctions.

III. Safety Supervisior Specification

Our approach for safeguarding against unpredictable behavior of intelligent functions is to specify a simple safety supervisor that detects malfunctions of the intelligent functions and triggers a minimum risk maneuver such as an emergency stop. In order to develop the safety supervisor according to traditional safety engineering approaches, explicit specification of the critical malfunctions is required. Our approach for deriving such a specification comprises four steps:

1. In the first step, we identify combinations of general behaviors like acceleration, coarse-grained situations like driving on highway, and related accidents like collision with a front vehicle.
2. In the second step, we derive concrete behavior malfunctions like "accelerating if a slower vehicle is closely in front".
3. In the third step, we formalize the malfunctions using constraints over parameters, such as "vx > 50km/h". We denote "vx" as a parameter and the condition "> 50km/h" as a constraint of that parameter.
4. In the last step, we estimate the risk of malfunctions and derive a related integrity level that determines with which level of confidence the malfunctions have to be detected and avoided.

In the following, we will explain and exemplify each step.

A. *Step 1: Identification of general behaviors, situations, and related accidents*

In the first step, we consider general behaviors in coarse-grained situations and reason on which accidents can occur. The first challenge in this context is that we need to cover all possible behaviors and situations. If there is any behavior or situation that is not covered by this analysis, we do not know whether or not an accident might potentially occur. The second challenge is to identify all potential accidents for each combination of behavior and situation.

To handle these challenges, we have to find a good abstraction level for the description of behaviors, situations, and accidents. If we use descriptions that are too fine-grained, the number of situations, behaviors, and accidents will explode and we will have non-viable effort. If we use descriptions that are too coarse-grained, we might miss too many details which are necessary for deriving malfunctions in the next step. In the following, we propose solutions with respect to the appropriate level of granularity.

Situation description: To describe a situation, we need complete partitioning of operational situations. To this end, we consider classes of operational situations. Thus, we need a complete set of classes of operational situations. Additionally, these classes should support the task of identifying accidents. In order to reason about possible accidents, the situation classes should produce a concrete picture in the safety engineer's mind. We therefore propose "location of use" as a class for partitioning an operational situation. For the automotive domain, we handled the challenge of obtaining a complete list of usage locations by reviewing and integrating ten industrial hazard analyses and risk assessments (HRAs) and a coordinated catalog of operational situations developed by five German automotive OEMs. Table 1 shows our complete list of usage locations. Please note that completeness is restricted to completeness with respect to these ten HRAs and the coordinated catalog. If any further lists or additional locations of use exist, the list can be extended with this information.

Location of use
Highway
Country road
City
Mountain pass
30 km/h zone
Construction site
Tunnel
Parking lot
Garage
Workshop
Roadside
Parking lane
Car wash

Table 1- "Complete" list of locations of use

Behavior description: To describe behaviors, we propose deriving these from the operator actions of non-autonomous vehicles because we can put ourselves in the position of the operator in order to identify accidents. Furthermore, we propose omitting any quantification such as accelerating at 5 m/s^2 and consider general behavior classes like steering left, steering right, accelerating, and decelerating. In this way, an explosion of behaviors can be avoided.

Accident description: Our rule for describing accidents is to describe them in as general terms as possible but in enough detail to estimate their severity. The reason is that the integrity level with which we have to safeguard against malfunctions causing an accident depends on the severity of the accident.

B. *Step 2: Malfunction identification*

In the second step, we derive malfunctions from the combination of behavior class, situation class (location of use), and accident class. If a behavior from the behavior class occurs in a situation from the situation class, then this is not necessarily a malfunction causing an accident. Accordingly, we try to find all combinations of behavior and situation that can cause an accident. To this end, we propose performing a fault-tree analysis for each combination of behavior class, situation class, and accident. The top-event of the fault tree is the accident. The basic events are combinations of behaviors and situations. The Boolean relation between the basic events and the top-event can be used to visualize deductive reasoning steps with respect to the selection of behaviors from the behavior class and the selection of situations from the situation class. The usage of a fault tree is obviously not mandatory and only makes sense if an accident has many malfunctions causing it.

For instance, we consider the combination of the general behavior "acceleration", the location of use "highway", and the general accident "collision with a front vehicle". The behavior "acceleration" is obviously not generally a malfunction if we drive on a highway. We thus reason about special causes where it is a malfunction and find the special case "accelerating if a slower vehicle is closely in front".

C. *Step 3: Malfunction formalization*

In the third step, we formalize the malfunctions by stating constraints over parameters, such as "vx > 50km/h". In the ideal case, the parameters are already determined by traditional functions of non-autonomous vehicles. In any case, they should be determinable by reliable (non-intelligent) functions because our goal is to build a simple safety supervisor that requires no artificial intelligence.

A parameter constraint is usually not only a formal representation of a textual description, because it quantifies things that need not be quantified in the textual description. Consequently, a parameter constraint is generally a refinement of a textual description. Typically, there are temporal dependencies between the possible refinements of the malfunctions. This is due to malfunctions being critical combinations of behavior and situation, and because a behavior changes the situation. If we continue with an unwanted behavior in a situation, then we get a new situation where the behavior is even more unwanted. This can continue until an accident occurs. If we have such temporal dependencies, then we have to decide at which point in time we want to forbid the behavior with which level of integrity. In this context, we can also select several points in time with different integrity levels and follow a "layer of protection" strategy.

We did, for instance, derive five constraints for the malfunction "accelerating if a slower vehicle is closely in front" – one for each criticality level in ISO 26262, ranging from lowest criticality (QM) to highest criticality (ASIL D). The first refers to malfunctions that only require QM to be safeguarded against because we are far away from an accident in terms of time and the distance to the front vehicle is very large. Due to the large distance, we have less reliable distance data in order to check if the constraint is fulfilled. Consequently, our confidence that our check will never fail is quite low. This does not matter because we have four other constraints as a fallback layer. These constraints refer to situations where the distance to the front vehicle is lower and hence criticality is higher (ASIL A to ASIL D). The constraint shown in Figure 1 captures malfunctions of the highest criticality (ASIL D). The constraint contains four parts connected by an AND operator. If these malfunctions occur, it is very likely that there will be a very severe collision. We derived the ASIL D constraint using the method presented in [2]. The fundamental idea of this method is to construct parameter constraints according to the different risk levels. The idea is implemented by partitioning the space of situations according to severity and controllability. Regarding the four parts of the constraint, the first part refers to the necessary condition for the occurrence, the second and third parts to the severity, and the fourth part to the controllability of the accident. However, the application of this method is not mandatory for implementing this methodological step.

$$\left(-\frac{\Delta v(t_{Sit})}{\Delta a(t_{Sit})}+\sqrt[2]{\frac{\Delta v(t_{Sit})^2}{\Delta a(t_{Sit})}+\frac{2d(t_{Sit})}{\Delta a(t_{Sit})}}\right)\leq 3\ s\ AND$$

$$\Delta v(t_{Sit})+\Delta a(t_{Sit})\left(-\frac{\Delta v(t_{Sit})}{\Delta a(t_{Sit})}+\sqrt[2]{\frac{\Delta v(t_{Sit})^2}{\Delta a(t_{Sit})}+\frac{2d(t_{Sit})}{\Delta a(t_{Sit})}}\right)> 1{,}4\frac{m}{s}\ AND$$

$$vx_1(t_{Sit})+ax_1(t_{Sit})\left(-\frac{\Delta v(t_{Sit})}{\Delta a(t_{Sit})}+\sqrt[2]{\frac{\Delta v(t_{Sit})^2}{\Delta a(t_{Sit})}+\frac{2d(t_{Sit})}{\Delta a(t_{Sit})}}\right)> 13{,}9\frac{m}{s}\ \ AND$$

$$\mathrm{d}(t_{Sit})<\frac{vx_1(t_{Sit})^2}{2ax_1}+0{,}5\ vx_1(t_{Sit})$$

Figure 1 – ASIL D parameter constraint

D. *Step 4: Integrity level determination*

In the previous step, we superficially considered the level of integrity with which we have to ensure the parameter constraints. In this step, we do this more systematically and for all parameter constraints.

As we are dealing with road vehicles, we adopt the approach of ISO 26262. In ISO 26262, an ASIL is derived from the risks of *hazardous events* and is attached to a safety goal referring to the hazardous events. A hazardous event is a behavior failure in a situation. For instance, the behavior failure "self-acceleration" in the situation "driving with high speed on a highway and a slower vehicle is closely in front" is a hazardous event. As the example shows, hazardous events are similar to our malfunctions and their formal representation as parameter constraints. The main difference between our malfunctions and hazardous events is that our malfunctions refer to autonomous behavior like "acceleration" instead of to failures of non-autonomous behavior like "self-acceleration". This difference, however, is not relevant for estimating risks and deriving ASILs. For instance, the risk of "self-acceleration" in a certain situation is obviously the same as the risk of "acceleration" by an intelligent function in the same situation. Accordingly, we adopt the risk estimation and the related integrity level derivation from ISO 26262 in order to determine ASILs for our parameter constraints.

IV. Safety Supervisior Implementation

The requirements specification for our safety supervisor comprises a set of parameter constraints. Each of these constraints captures a set of similar malfunctions that have to be safeguarded against with the same integrity level.

For safeguarding against the malfunctions, the safety supervisor has to monitor each parameter constraint and the planned behavior of the intelligent function. If the planned behavior might change the current values of the parameters so that a malfunction might occur, then the safety supervisor must detect this, prevent the intelligent function from performing the behavior, and trigger a minimum risk maneuver. In this section, we propose solutions for the following two problems with respect to the detection of malfunctions:

1. Omission of malfunction detection

If the determined value of a parameter deviates from the actual value, then a parameter constraint might evaluate to *false* instead of to *true.* Consequently, a malfunction might not be detected. The integrity level with which such an omission failure has to be avoided is given by the integrity level of the parameter constraint that evaluates to *false* instead of to *true*.

2. Commission of malfunction detection

The complementary failure is that a constraint evaluates to *true* instead of to *false*. Consequently, a malfunction might be detected even though there is no malfunction, and a minimum risk maneuver will be performed unnecessarily. As minimum risk maneuvers can be dangerous maneuvers, we have to minimize the cases where they are performed unnecessarily.

Our fundamental idea for handling these problems is to give guarantees for the determination of parameter values and to adapt the parameter constraints according to these guarantees and the avoidance of omission failures.

For instance, if we guarantee that the distance parameter shown in Figure 1 will be maximally 5 meters too high, then we change the parameter constraint so that it will evaluate to *true* even if the determined distance is 0-5 meters too high.

As the example shows, we avoid omission failures by accepting some commission failures. The number of commission failures decreases with the quality of the guarantees. Considering the example above, we can, for instance, minimize commission failures by guaranteeing a limit of maximally 2 meters instead of 5 meters.

For this reason, we want to give the strongest guarantees. However, it can be very effort-intensive to find the strongest guarantees and sometimes a stronger guarantee does not reduce the number of commission failures. Therefore, we developed an approach that enables us to analyze the benefits of a stronger guarantee before investing effort into checking if it is feasible to give this strong guarantee. In the following, we will explain and exemplify the steps of our solution approach.

A. *Remove parameters whose integrity level is too low*

In the first step, we get rid of parameters for which we cannot give any guarantees with the integrity level that is attached to the parameter constraint. If we identified such a parameter, then we have to come up with a new parameter constraint that does not contain the parameter and that has a certain relationship to the original parameter constraint.

Informally, the relationship between the original constraint and the new constraint can be described via the space of concrete malfunctions that is captured by the constraints. The new space must comprise all malfunctions of the original space and should also minimize the number of added malfunctions.

Formally, this means that we search for a weaker condition that does not contain the parameter. If there are several weaker conditions, then we search for the strongest because we want to minimize the number of additional malfunctions.

For instance, we consider the ASIL D constraint shown in Figure 1 and assume that we cannot determine the acceleration and the speed of the front vehicle with ASIL D. Consequently, we cannot determine the parameters for the differences between the parameters "$\Delta v(t_{Sit})$" and "$\Delta a(t_{Sit})$" in the ASIL D constraint. We thus remove the first three parts of the constraint and get the following new constraint, which comprises no "delta parameters":

$$\mathrm{d}(t_{Sit}) < \frac{vx_1(t_{Sit})^2}{2ax_1} + 0{,}5\ vx_1(t_{Sit})$$

Figure 2 – Remaining ASIL D parameter constraint

This constraint is weaker than the one shown in Figure 1 and thus captures all original malfunctions.

B. *Identifying causes for omissions and commissions*

In the second step, we analyze which failures of the parameter determination can cause an omission of malfunction detection, resp. a commission of malfunction detection.

For instance, we consider the constraint shown in Figure 2 and the distance parameter. If the distance is determined as being too high, then the constraint might evaluate to *false* instead of to *true*. We might thus think that there is no malfunction even though there is one. Consequently, a "too high" failure of the distance parameter can cause an omission of malfunction detection. A "too low" failure can only cause a commission of malfunction detection.

C. *Safeguarding against causes for comissions*

In the third step, we establish guarantees for the failures that can cause commissions of malfunction detection.

Which guarantees are required obviously depends on the concrete minimum risk maneuver that would be performed unnecessarily. We thus derive the required guarantees from the minimum risk maneuvers and then check if we can assure these guarantees. If we cannot assure them, we have to remove the related constraints. This means we have to reduce the number of malfunctions against which our approach safeguards. Consequently, there must be some other kind of safeguard against these malfunctions, e.g., validation of the intelligent function.

D. *Safeguarding against causes for omission*

In the fourth step, we derive guarantees for the failures that can cause omissions of malfunction detection and adapt the parameter constraints according to these guarantees.

As explained above, we want to give the strongest guarantees in order to minimize commission failures due to the adaptation of the parameter constraints. As it is often very effort-intensive to find the strongest guarantees and as a stronger guarantee does not always reduce the number of commission failures, we parameterize the guarantees and evaluate different parameter settings by means of visualization.

We visualize all constraints as spaces (of malfunctions) and evaluate how they grow and shrink if we modify the guarantees. In particular, we evaluate if the union of all spaces grows, because this means that the number of commissions increases. Furthermore, we evaluate how the different spaces overlap, because if a space of higher integrity subsumes a space of lower integrity, then the subsumed space is already sufficiently safeguarded. This means that the safety supervisor only needs to monitor the constraint of the higher integrity.

For instance, we consider the inequality in Figure 2. We transform this inequality into the following equation:

$$\mathrm{d}(t_{Sit}) - \Delta\mathrm{d} = \frac{(vx_1(t_{Sit}) + \Delta v)^2}{2ax_1} + 0{,}5\ (vx_1(t_{Sit}) + \Delta v)$$

Figure 3 – Equation for visualizing the ASIL D space

The parameters "Δd" and "Δv" refer to the guarantee that we give for a "too high" failure of the distance and a "too low" failure of the speed of our car. We replaced the less-than sign with an equal sign because we do not visualize the space of malfunctioning behavior directly but rather the border between malfunctioning behavior and non-malfunctioning behavior. The line at the top of Figure 4 shows this border for a certain parameter setting of "Δd" and "Δv". The other two lines refer to the other constraints we derived in III.C.

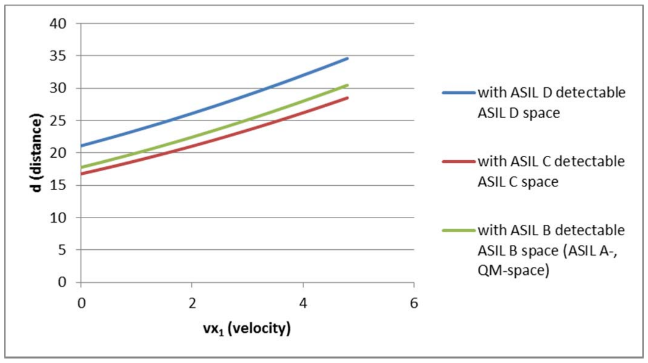

Figure 4 – Illustration of ASIL spaces

As can be seen, the QM constraint, the ASIL A constraint, and the ASIL B constraint are represented by the same line. The reason for this is that the ASIL B constraint implies the ASIL A and QM constraints after the removal of parameters whose integrity level is too low (cf. subsection IV.A). If we take the guarantees for "Δd" and "Δv" that are visualized in Figure 4, then the maximal distance where acceleration is allowed is always given by the ASIL D space. Consequently, the other ASIL spaces need not be monitored by the safety supervisor if we use the guarantees.

As the example shows, this visualization supports not only the identification of reasonable guarantees but also tells us which constraints we need to monitor with our safety supervisor. This is also very important information that can reduce the implementation effort significantly.

V. Related Work

The most closely related work is our previous work presented in [2], where we aimed at a specification that explicitly describes all critical situations for a given behavior, like acceleration. We revised this approach for two reasons. First, we recognized in case studies that it is often almost impossible to specify all critical situations. Second, we recognized that the parameter constraints are typically descriptions of unwanted vehicle behavior in a situation instead of pure situation descriptions. However, the main contribution to this previous work is the implementation approach described in Section IV. Another contribution is the justification of our approach described in Section II.

In [3], situations are formalized via parameter constraints in order to ensure that a behavior failure is evaluated in all possible situations that can occur at runtime. We adopted the formalization approach in order to specify critical combinations of behavior and situation.

In [4], an approach is presented for a safety-oriented reference architecture that introduces adaptive software safety cages. However, it is neither explained how malfunctions of intelligent functions can be specified explicitly nor how such malfunctions can be detected with an acceptable integrity level.

In [5], the authors investigate what has to be considered in order to achieve safe autonomous driving. Regarding the safety concept for autonomous driving, they refer to [6] where several actions like “limp home”, “safe parking on the next parking place”, “immediate stopping on the roadside”, and “immediate emergency brake” are listed. According to [7], these actions are also reasonable for safeguarding fully autonomous driving. This supports our fundamental assumption that there are some minimum risk maneuvers that can be applied. The identification of reasonable maneuvers complements our work and is strongly related because any detection of an unwanted behavior in a situation is only beneficial if a better alternative behavior is known that minimizes risks.

The functional safety standard “ISO 26262 - Road Vehicles - Functional Safety” is obviously related to our work because we adopted some of its concepts. ISO 26262 considers malfunctions of non-intelligent functions. These malfunctions differ from our malfunctions as they refer to behavior failures that are deviations from the behavior that should be generated by the non-intelligent functions. We defined malfunctioning behavior as a combination of behavior and situation. This definition is related to the definition of a hazardous situation, that is, a behavior failure in a certain situation. In accordance with this similarity, we adopted the risk estimation approach of ISO 26262 if our approach is applied to road vehicles. Furthermore, we adopted the derivation of the integrity levels from the estimated risk because we believe that the recommendations of ISO 26262 fit quite well for the development of a safety supervisor.

VI. Summary and Conclusion

In this paper, we investigated problems related to the application of traditional safety engineering for autonomous vehicles. We focused on the problem that autonomy is achieved by intelligent functions that generate unpredictable behavior. Unpredictability is a problem for safety assurance since verification and other important tasks require explicit specification of the behavior and are thus not applicable. To overcome this problem, our approach allows explicitly specifying some critical combinations of vehicle behavior and runtime situation and safeguarding against them.

Our safety engineering approach clearly distinguishes between specification of critical behaviors in runtime situations and avoidance of the specified behaviors in these situations. We conclude that this clear distinction might makes sense not only for intelligent functions that generate unpredictable and autonomous behavior, but also for functions and behaviors that are just too complex to be tackled efficiently and reasonably by means of traditional safety engineering – or even for today’s vehicles to safeguard against erroneous and potentially dangerous operator inputs. Considering the latter case, a vehicle should, for instance, follow control commands only if all risks are acceptable. This means that if an operator controls a road vehicle such that it would skid off the road, then the vehicle should not follow the control commands. As regards very complex functions, the explicit, correct, and complete specification of their behavior is typically a challenging task and mishaps and systematic specification faults are likely. In this case, if safety engineering focuses on behavior failures only, that is, deviations from the specified behavior, then such specification faults are probably not addressed sufficiently.

Acknowledgment

The work presented in this paper was created in context of the Fraunhofer Digital Commercial Vehicle Technology Cluster, which was funded by the State of Rhineland-Palatinate and the Fraunhofer-Gesellschaft.

References

[1] ISO 26262: Road Vehicles, Functional Safety Part 1 to 10 (2012)

[2] R. Adler, S. Kemmann, “Towards safe autonomic driving. Enhancing traditional hazard and risk analysis” in proceedings of the 3rd Commercial Vehicle Technology Symposuim (CVT'2014), Shaker, 2014, 3-12.

[3] Kemmann, S. and Trapp, M., "SAHARA -A Systematic Approach for Hazard Analysis and Risk Assessment," SAE Technical Paper 2011-01-1003, 2011, doi:10.4271/2011-01-1003.

[4] K. Heckemann, M. Gesell, T. Pfister, K. Berns, K. Schneider, M. Trapp, "Safe Automotive Software", in proceedings of "Knowledge-Based and Intelligent Information and Engineering Systems: 15th International Conference, KES 2011, Kaiserslautern, Germany, September 12-14, 2011, Proceedings, Part IV", Springer, pp. 167-176,

[5] H. Winner, M. Maurer: Sicherheit. in „Autonomes Fahren. Technische, rechtliche und gesellschaftliche Aspekte“ Wiesbaden: Springer Vieweg. pp. 417-538

[6] Binfet-Kull, M., Heitmann, P., Ameling, C. (1998). System safety for an autonomous vehicle. 1998 IEEE Intelligent Vehicles Symposium (IV), Stuttgart, Germany.

[7] Gasser, T. M., Arzt, C., Ayoubi, M., Bartels, A., Bürkle, L., Eier, J., Flemisch, F., Häcker, D., Hesse, T., Huber, W., Lotz, C., Maurer, M., Ruth-Schumacher, S., Schwarz, J., Vogt, W. (2012). „Rechtsfolgen zunehmender Fahrzeugautomatisierung: gemeinsamer Schlussbericht der Projektgruppe“ Wirtschaftsverlag NW Verlag für neue Wissenschaft