Performing a More Realistic and Complete Safety Analysis by Means of the Six-Variable Model

Nelufar Ulfat-Bunyadi[1] Denis Hatebur[2] Maritta Heisel[3]

Abstract: Safety analysis typically consists of hazard analysis and risk assessment (HARA) as well as fault tree analysis (FTA). During the first, possible hazardous events are identified. During the latter, failure events that can lead to a hazardous event are identified. Usually, the focus of FTA is on identifying failure events within the system. However, a hazardous event may also occur due to invalid assumptions about the system's environment. If the possibility that environmental assumptions turn invalid is considered during safety analysis, a more realistic and complete safety analysis is performed than without considering them. Yet, a major challenge consists in eliciting first the 'real' environmental assumptions. Developers do not always document assumptions, and often they are not aware of the assumptions they make. In previous work, we defined the Six-Variable Model which provides support in making the 'real' environmental assumptions explicit. In this paper, we define a safety analysis method based on the Six-Variable Model. The benefit of our method is that we make the environmental assumptions explicit and consider them in safety analysis. In this way, assumptions that are too strong and too risky can be identified and weakened or abandoned if necessary.

Keywords: safety analysis, hazard analysis, risk analysis, fault tree analysis, assumption, environment, six-variable model

1 Introduction

The focus of FTA is usually on identifying the failure events within the system that can lead to a hazardous event (cf. e.g. [IS11], [RS15]). Yet, a system is always embedded in an environment. It is designed for this environment and satisfies its requirements during operation only if the assumptions about this environment that were made during development are valid [ZJ97]. A hazardous event may also occur due to environmental assumptions that turn out to be invalid. Van Lamsweerde describes in his book [La09] several accidents in which wrong or invalid assumptions about the environment led to a hazardous event, while the system behaved as specified. A famous example is the Lufthansa A320 flight to Warsaw. The plane ran off the end of the waterlogged runway resulting in injuries and loss of life. The reverse thrust was disabled for up to nine seconds after landing. The reason was the following. The autopilot had the task to enable reverse thrust only if the plane is moving on the runway. Since the wheels were not turning due to aquaplaning, the autopilot 'assumed' that the plane is not moving on the runway and thus disabled reverse thrust. So, the system behaved as specified but the hazardous event occurred nevertheless

[1] University of Duisburg-Essen, Oststrasse 99, 47057 Duisburg, nelufar.ulfat-bunyadi@uni-due.de
[2] Institut für technische Systeme GmbH, Emil-Figge-Str. 76, 44227 Dortmund, d.hatebur@itesys.de
[3] University of Duisburg-Essen, Oststrasse 99, 47057 Duisburg, maritta.heisel@uni-due.de

due to the wrong assumption. Due to such accidents, we argue that safety analysis should not only focus on hazardous events resulting from failure events within the system. Wrong or invalid environmental assumptions should also be considered as possible causes of a hazardous event. This will result in a more realistic and more complete safety analysis. To realize that it is necessary (i) to make the environmental assumptions explicit and (ii) to take the probability for their invalidity into account in safety analysis. In previous work, we developed the Six-Variable Model which provides the required support in making the environmental assumptions explicit. In this paper, we present a safety analysis method that is based on the Six-Variable Model and extends traditional safety analysis in order to consider also invalid environmental assumptions as possible causes of hazardous events.

The paper is structured as follows. In Section 2, we first introduce the Six-Variable Model. In Section 3, we briefly describe some fundamentals that we use. Our extensions to safety analysis are then presented in Section 4. To demonstrate that our method results in a more realistic safety analysis, we compare our method to traditional safety analysis in Section 5. Finally, we discuss related work in Section 6 and draw conclusions in Section 7.

2 The Six-Variable Model

The problem with assumptions is that developers do not always document the assumptions they make. Frequently, they are even not aware of them. Even if they are aware of the importance to document assumptions, they are left alone with the question which information to document. Our Six-Variable Model provides support in this regard.

The Six-Variable Model [UBMH16] is based on the well-known Four-Variable Model [PM95] and focuses on control systems. A control system consists of some control software which uses sensors and actuators to monitor/control certain quantities in the environment. In vehicles and air planes, we find a lot of control systems, e.g. ACC (Adaptive Cruise Control) or ESP (Electronic Stability Program). The Four-Variable Model defines the content of software documentation for control systems. Therein, Parnas and Madey differentiate between four variables. Monitored variables m are environmental quantities the control software monitors through input devices. Controlled variables c are environmental quantities the control software controls through output devices. Input variables i are data items that the control software needs as input and output variables o are quantities that the control software produces as output.

However, frequently, it is not possible to monitor/control exactly those variables one is interested in. Instead, a different set of variables is monitored/controlled, whose variables are related to the ones of real interest. The Six-Variable Model demands that the variables of real interest should be documented as well (beside the classical four variables). In addition, it should also be documented how the variables of real interest are related to the monitored/controlled ones. The Six-Variable Model is depicted in Figure 1 as a problem diagram. We first explain the notation and then the content of the model.

Problem diagrams have been introduced by Jackson [Ja01]. A problem diagram shows the so called machine (i.e. the software-to-be), its environment, and the requirement to be sat-

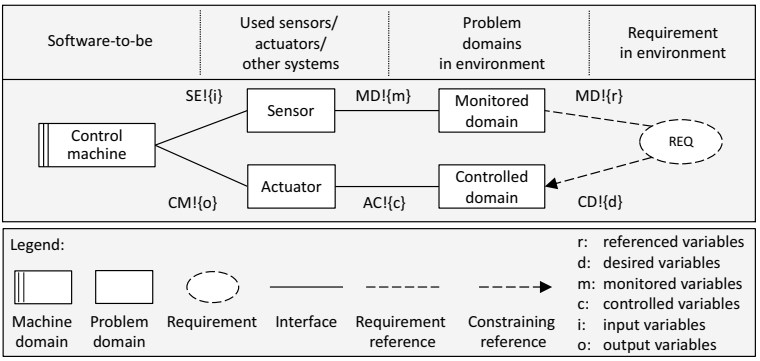


Fig. 1: The Six-Variable Model [UBMH16]

isfied. A problem diagram contains the following modelling elements: a machine domain, problem domains, a requirement, interfaces, requirement references, and constraining references. A problem domain represents any material or immaterial object in the machine's environment. The requirement is to be satisfied by the machine together with the problem domains. Interfaces exist between machine domain and problem domains or among problem domains. At the interfaces, phenomena (e.g. events, states, values) are shared. Sharing means that one domain controls a phenomenon, while the other observes it. At an interface, not only the phenomena are annotated but also an abbreviation of the domain controlling them followed by an exclamation mark (e.g. CM!). A requirement is connected to problem domains by means of a requirement reference or a constraining reference. A requirement reference expresses that the requirement refers somehow to the domain phenomena, while a constraining reference expresses that the requirement constrains (i.e. influences) them.

In the Six-Variable Model (Figure 1), the machine is the control software. It uses sensors and actuators to monitor/control certain phenomena of environmental domains (*m* and *c* variables). The requirement is shown on the right-hand side of the model. It refers to and constrains certain phenomena of the environmental domains. As stated above, these phenomena are not necessarily the same phenomena as the controlled/monitored ones. We call these phenomena the *r* and *d* variables. *r* (referenced) variables are environmental quantities that should originally be observed in the environment. Originally means before deciding which sensors/actuators/other systems to use for monitoring/controlling. As explained above, this decision frequently results in a different set of variables which are monitored/controlled. *d* (desired) variables are environmental quantities that should originally be influenced in the environment.

The benefit of making the six variables explicit is, among others, that we can make the environmental assumptions explicit based on them. Usually, we are the developers of the machine but the sensors/actuators are developed by other parties. We depend on them for satisfying the requirement *REQ* (see Figure 1). We assume, for example, that the sensors provide an *i* variable which actually reflects the corresponding *m* variable. Figure 2 shows the environmental assumptions for the Six-Variable Model. We differentiate between two

types of assumptions (based on [La09]): domain hypotheses and expectations. A domain hypothesis is a descriptive statement about the environment which needs to be valid. An expectation is a prescriptive statement to be satisfied by an environmental agent like a person, sensor, actuator. So, *DH-MD* is a hypothesis about the monitored domain, which needs to be true. *Exp-SE* is an expectation to be satisfied by the sensors, *Exp-AC* is an expectation to be satisfied by the actuators, and *Exp-CD* is an expectation to be satisfied by the controlled domain. *SOF* represents the software requirements which are to be satisfied by the control machine. The requirements *REQ* can only be satisfied, if *DH-MD* is valid and *Exp-SE, SOF, Exp-AC* as well as *Exp-CD* are satisfied.

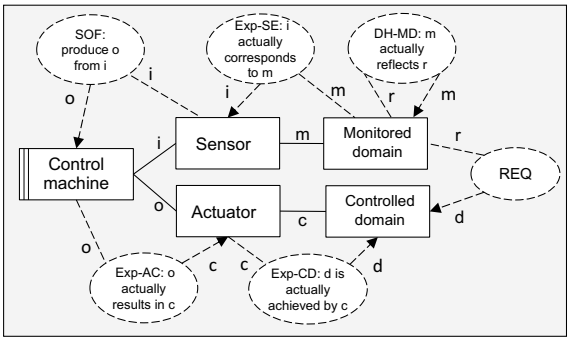


Fig. 2: Assumptions regarding the six variables [UBMH16]

Note that there may be several connection domains (i.e. a chain of sensors or a chain of actuators) between the machine and the environmental domains (Figure 1), especially in embedded systems. The existence of connection domains means that there are not only six variables to be documented but even $4+n$ variables. However, our Six-Variable Model supports that (see [UBMH16] for more details).

In case of the Lufthansa air plane (see Figure 3), for example, a referenced variable is the plane's movement on the runway. This phenomenon was observed by monitoring the turning of the wheels. A desired variable was the deceleration of the plane. This should be achieved by enabling the reverse thrust (controlled variable). Note that we show the same plane domain twice in Figure 3 and annotated it with an asterisk. Usually, this is not allowed in problem diagrams (i.e. a domain is only shown once in a diagram) but we did it here to enable the reader to see the similarity to Figure 1. Figure 3 shows also the environmental assumptions for the Lufthansa autopilot. Unfortunately, such a model was not created for the autopilot. Yet, making environmental assumptions explicit helps already since developers then start reflecting on them. However, we go one step further and ask for estimating the probability for their invalidity and taking these values into account in safety analysis. If it then turns out that assumptions are too strong and too risky, they can be weakened or abandoned. In this way, hazardous events resulting from invalid environmental assumptions can be prevented.

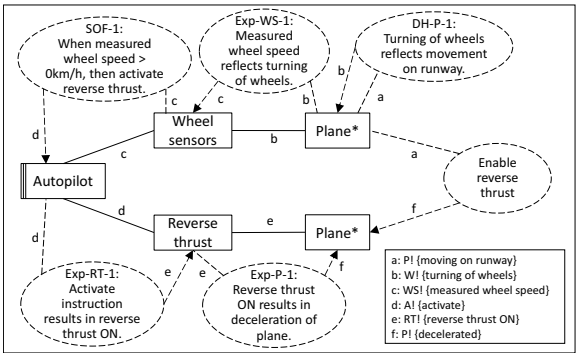


Fig. 3: Six variables in case of the Lufthansa autopilot

3 Fundamentals

Beckers et al. [Be13] have developed a hazard analysis and risk assessment (HARA) method which is compliant to the ISO 26262 Standard [IS11]. We use and extend this method in Section 4. Therefore, we introduce it here briefly. The goal of the HARA method is to identify potential hazards for the considered system and to formulate safety goals related to the prevention or mitigation of these hazards in order to achieve an acceptable residual risk. The method consists of seven steps. First, a so called context diagram[4] is created and a list of high-level requirements is defined. Second, for each high-level requirement, possible faults are identified using certain guide words (e.g. no, unintended, early, late) as a support. Third, for each high-level requirement, situations are selected from a given list of possible situations (during a vehicle's life) that are considered to be relevant for the requirement (e.g. parking manoeuvre, braking situation, highway situation). Fourth, for each fault/requirement combination, all situations that could lead to a potential hazard are identified in the list of situations being relevant (result from Step 3). Fifth, each hazard is classified according to its severity, exposure, and controllability in order to assess the required level of risk reduction (ASIL). The possible ASILs are: QM, ASIL A, ASIL B, ASIL C, and ASIL D. ASIL D is the highest level requiring the highest risk reduction and QM is the lowest level expressing that the normal quality measures applied in the automotive industry are sufficient. Sixth, safety goals are defined to address the hazards (i.e. to prevent or mitigate them). Seventh, the results of the hazard analysis and risk assessment are reviewed by an independent party.

4 Our Safety Analysis Method

Traditional safety analysis focuses on the system (cf. e.g. [IS11]). The system is defined as a set of elements that relates at least a sensor, a controller and an actuator with one another [IS11]. During traditional safety analysis, hazardous events occurring at the actuators are

[4] Context diagrams have also been introduced by Jackson [Ja01]. A context diagram is similar to a problem diagram but it shows only the machine and the problem domains connected to it.

identified (see Figure 1) and their causes are analysed until arriving at the sensors. Possible causes of a hazardous event (i.e. *c* variable is not achieved/effected) are then: a fault in the actuators, a fault in the machine, or a fault in the sensors.

Yet, actually, hazardous events occur at the controlled domains in the environment (see Figure 1). For example, in case of the Lufthansa air plane, the controlled variable was 'reverse thrust ON'. Yet, the desired variable was 'plane decelerated' (see Figure 3). This is what we actually wanted to achieve in the environment. During HARA, we should therefore identify situations in which this goal is not achieved (i.e. *d* variable is not achieved). Such hazardous events represent situations, in which assumptions of type *Exp-CD* (see Figure 2) turn out invalid. Furthermore, FTA should not stop at the sensors because there are further possible causes for a hazardous event. As pointed out in Section 2, we make the assumption that the monitored variables *m* reflect the referenced variables *r* (assumptions of type *DH-MD* in Figure 2). Yet, these are assumptions that can also turn invalid. Actually, this was the case in the Lufthansa example. The turning of the wheels did not reflect the plane's actual movement on the runway in that situation, i.e. *DH-P-1* (see Figure 3) turned out invalid. We need to consider the possibility that such assumptions turn invalid in the FTA. Therefore, FTA should not stop at the sensors but at the monitored domains. Further possible causes of a hazardous event are then: a 'fault' in the monitored domain, i.e. a wrong/invalid domain hypothesis.

In the following, we describe a safety analysis method that performs HARA and FTA in the way we just suggested. Our method is compliant to the ISO 26262 Standard [IS11]. It consists of the five steps depicted in Figure 4. The figure provides an overview of the method steps as well as inputs and outputs of each step. They are explained in the following.

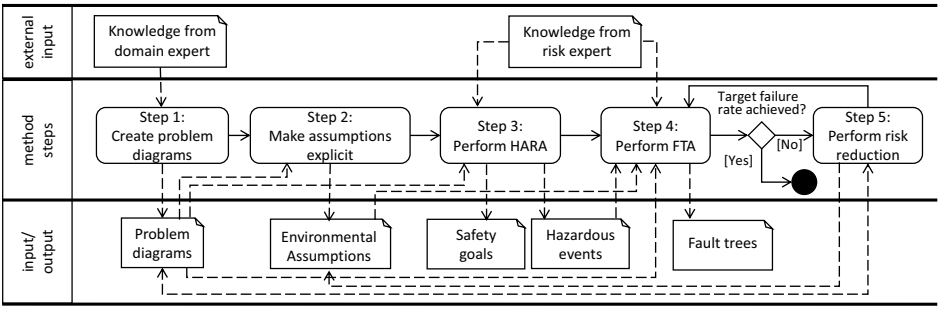


Fig. 4: Overview of our method

Step 1: Create problem diagrams. As a first step, one or (if necessary) several problem diagrams need to be created for the system's main functionality based on our Six-Variable Model presented in Section 2. In the ISO 26262 Standard, the creation of a functional architecture showing the main system components is suggested. Instead of the functional architecture, we suggest using problem diagrams as a foundation for HARA because they show not only the system but also the environment. For the problem diagrams it is essentially important that all domains connecting the machine to the real world are actually modelled in the problem diagrams even if they only transmit data without processing it

(e.g. in case of a chain of sensors or a chain of actuators). The reason is that failure events could also occur here.

Step 2: Make assumptions explicit. For each problem diagram from Step 1, the assumptions need to be made explicit as shown in Figure 2. Making the assumptions explicit has the benefit that we can assess the probability for their invalidity and consider them in the FTA (Step 4).

Step 3: Perform HARA. During this step, the HARA is mainly performed in the way described by Beckers et al. in [Be13] except that we identify hazardous events at the controlled domains (i.e. situations in which d variable is not achieved) and not at the actuators. Input to this step are the problem diagrams from Step 1. For each requirement in a problem diagram, possible faults are identified. For each fault/requirement combination, all situations that could lead to a potential hazard are identified. Each hazard is classified according to its severity, exposure, and controllability in order to assess the required level of risk reduction (ASIL). Finally, safety goals are defined for preventing or mitigating the hazards. In order to evaluate whether the residual risk of safety goal violations is sufficiently low, the ISO 26262 Standard requires for a system with ASIL C or ASIL D to determine the PMHF (cf. [IS11]). PMHF stands for 'Probabilistic Metric for random Hardware Failures' and is used to evaluate the probability of violation of the considered safety goal using, for example, quantified FTA and to compare the result of this quantification with a target value. This is what we do in Step 4.

Step 4: Perform FTA. For each hazardous event, we identify possible causes and document the failure events and faults in a fault tree based on the corresponding problem diagram from Step 1. The hazardous event occurs at the controlled domain in the problem diagram. To identify faults, we traverse the problem diagram, starting at this controlled domain and stopping at the monitored domain(s). Note that it might be necessary to traverse the same problem diagram for several hazardous events. We document the identified failure events and faults in a fault tree. A fault tree (FT) is a directed acyclic graph with two types of nodes: events and gates [RS15]. The notation is shown in Figure 6. The event at the top of a fault tree is the hazardous event being analysed. Events that are not further decomposed are basic events, i.e. either faults or situations. Gates represent how failures propagate, i.e. how failures can combine to cause a hazardous event. There are two types of gates. An AND gate means that the output event occurs if all of the input events occur. An OR gate means that the output event occurs if any of the input events occurs.

The fault tree allows one to calculate the probabilities of failure occurrence. We calculate the probability that a failure occurs in one hour of operation time, assuming that this is the time of a typical drive cycle (see [IS11], Part 5, Section 9.4.2.3, Note 2). We also assume a linear failure rate distribution. Therefore, our values for the probabilities in the fault tree and for the failure rates are the same. Failure rates are annotated at the leaves of the tree (based on expert knowledge) and are calculated bottom up to the root according to FTA rules for AND and OR gates (cf. [RS15]). As stated above, the resulting value (i.e. the actually achieved failure rate) must be compared to a target value. One possibility to obtain PMHF target values is given by means of Table 6 in Part 5 of the ISO 26262 Standard [IS11]. It requires for ASIL D a failure rate smaller than $10^{-8}h^{-1}$ and for ASIL C a failure

rate smaller than $10^{-7}h^{-1}$. Note that these PMHF target values only cover the hardware parts of the system. The ISO 26262 Standard does not consider faults that occur in the environment. Since we think that the environment should be considered, we would like the environment failure rate to have the same value as the PMHF target value. This means that the overall maximum failure rate for violating the safety goal would be $2 \cdot 10^{-8}h^{-1}$ for ASIL D and $2 \cdot 10^{-7}h^{-1}$ for ASIL C.

To use the same fault tree for calculating the actually achieved failure rate (in order to show compliance to ISO 26262), we set the failure rates of the environment events to 1, if they are connected with an AND gate, and to 0, if they are connected with an OR gate in order to ignore them. The actually achieved failure rate is compared to the target value (i.e. $2 \cdot 10^{-8}h^{-1}$ for ASIL D or $2 \cdot 10^{-7}h^{-1}$ for ASIL C). Only if the calculated value for the tree is higher than the target value, Step 5 needs to be performed.

Step 5: Perform risk reduction. During this step, modifications are made to the problem diagrams from Step 1, for example, a sensor is added to increase reliability of monitored information, or a different procedure is considered (e.g. calculation or estimation). By means of such modifications, the assumptions of type *DH-MD, Exp-SE, Exp-AC, Exp-CD* may change as well, i.e. they are strengthened, weakened, or abandoned. Based on these modified problem diagrams, Step 4 is performed again. This is done until the target failure rate is achieved.

5 Comparison of Traditional FTA and Our Method

To illustrate that performing safety analysis based on the Six-Variable Model (as we do in our method) results in a more realistic and complete safety analysis, we compare two types of fault trees: Fault Tree 1, which focusses on the system (result of traditional FTA) and Fault Tree 2, which considers also the environment with the *r* and *d* variables. As an example we consider the ACC system described in [Ro03]. This is a simple version of an ACC, which mainly supports cruise control. It maintains the desired speed entered by the driver. If it detects vehicles ahead, it adapts the speed of the ACC vehicle accordingly.

Performing traditional FTA. The system design for the ACC is given as a SysML internal block diagram in Figure 5. The ACC uses data from a long range radar sensor (LRR) and ESP sensors to identify vehicles ahead on the same lane. For adapting the speed of the ACC vehicle, it uses the ESP system and the engine management system.

One hazardous event that may occur at the ESP (actuator) is *unintended braking*. We perform traditional FTA, i.e. we start analysis at the ESP and stop at the sensors (LRR and ESP sensors) focussing on the identification of system failures. The result, Fault Tree 1, is given in Figure 6. The hazardous event *unintended braking* may be caused by two events: the ESP system performs deceleration on its own (F ESP DEC) or it performs deceleration due to the input it received (FA ESP DEC). The first event represents a fault in the ESP system and is not further analysed. The second event may have different causes and is therefore further analysed. First, the CAN bus may have requested erroneously to decelerate, i.e. there was a CAN fault, (F CAN DEC1) or CAN requests for deceleration due

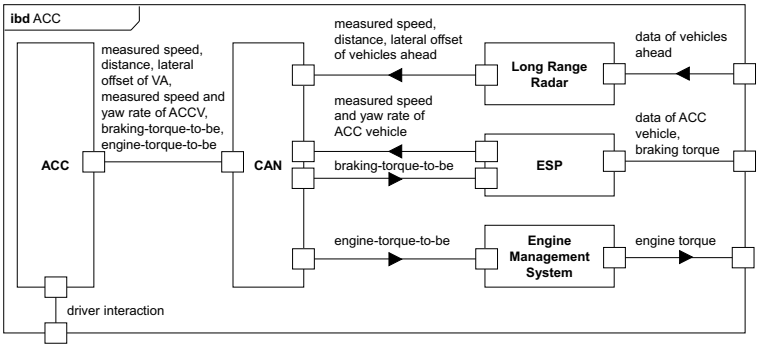


Fig. 5: ACC system design

to the input it receives from ACC (FA CAN DEC1). The latter event has again two possible causes: an ACC fault (F ACC DEC) or ACC requests deceleration due to the input it receives from CAN (FA ACC DEC). The latter event has again two possible causes: a CAN fault (F CAN DEC2) or CAN requests deceleration due to the input it receives from sensors. The latter event has again two possible causes: the ESP sensors send wrong information (wrong speed and/or wrong yaw rate) due to a fault in the ESP sensors or the long range radar sends wrong information (wrong speed and/or distance and/or lateral offset) due to a fault in the long range radar sensor.

Once the fault tree is created, failure rates must be assigned to the leaves of the fault tree and the failure rates must be calculated bottom up. Due to the experience one of the authors had in the automotive domain, the values were assigned by him in our example. We used the tool Reliability Workbench 11[5] to calculate the failure rates bottom up. To check the values calculated by the tool, we calculated them also manually. Overall, we achieve a failure rate of $2.04 \cdot 10^{-6} h^{-1}$.

Application of our method. We create first a problem diagram based on the Six-Variable Model. This is given in Figure 7. In contrast to Figure 5, Figure 7 shows the environment and the r and d variables. Referenced variables are speed, distance, and lane of vehicles ahead as well as the lane of the ACC vehicle. Monitored variables are speed, distance, and relative position of vehicles ahead as well as the course of the ACC vehicle. The desired variable is the speed adaptation of the ACC vehicle. The requirement in Figure 7 is 'Maintain desired speed and keep safety distance to vehicles ahead'. This requirement can be decomposed into the assumptions (expectations and domain hypotheses) and software requirements given below the problem diagram in Figure 7.

Instead of the hazardous event *unintended braking*, we consider the hazardous event that occurs at the controlled domain: *unintended speed reduction*. Based on the problem diagram in Figure 7, we create the fault tree. It is given in Figure 8. Compared to Fault Tree 1, Fault Tree 2 starts with a different root node due to the different hazardous event that is considered. This event has two possible causes. First, the ACC vehicle may decelerate

[5] http://www.isograph.com/software/reliability-workbench/

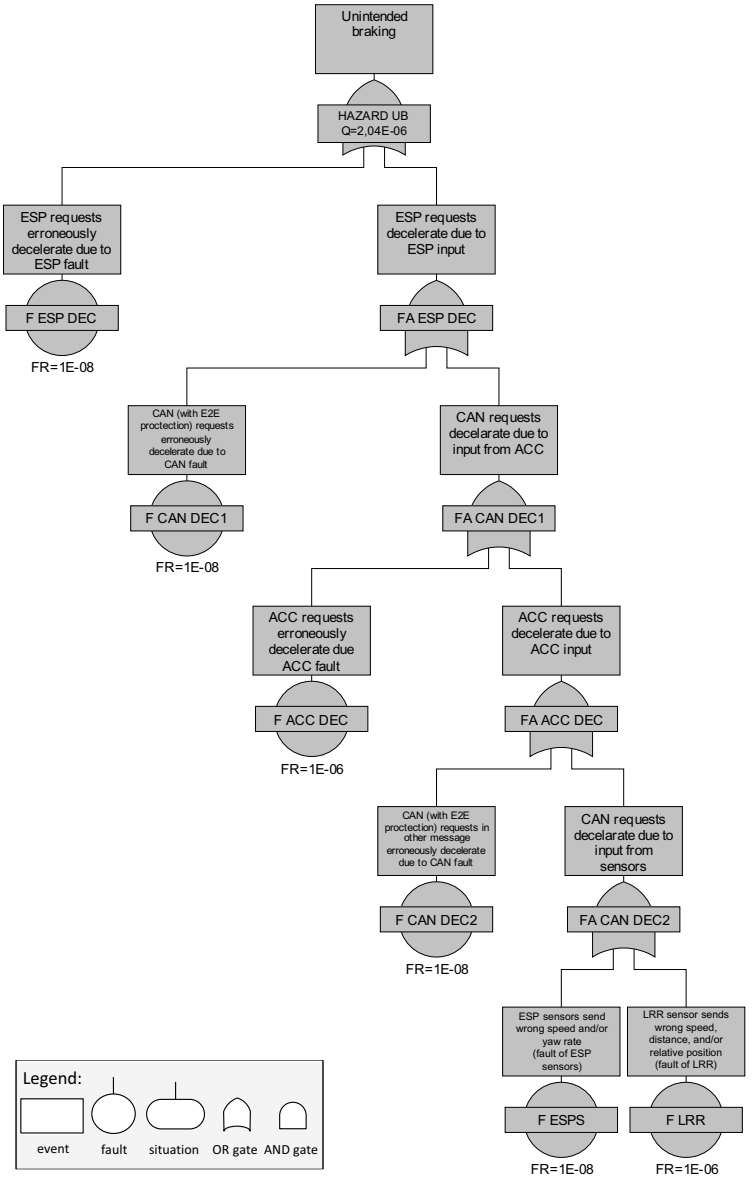
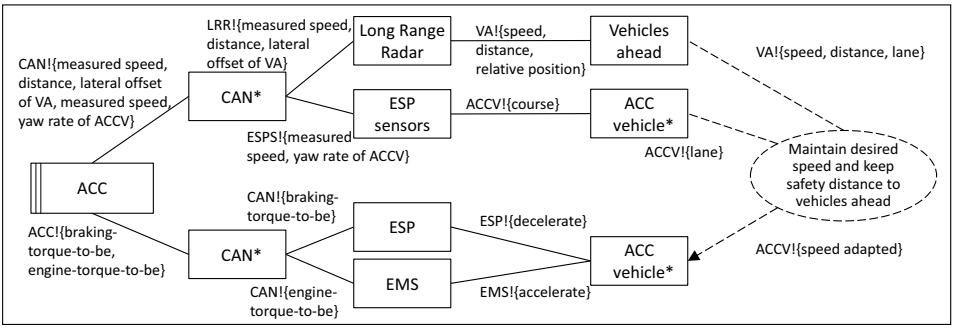


Fig. 6: Fault Tree 1 – Result of Traditional FTA



DH1-ACCV:	The course of the ACC vehicle reflects its lane.
DH1-VA:	Speed, distance, and relative position of a vehicle ahead reflect its speed, distance, and lane.
Exp1-LRR:	Measured speed, distance, and lateral offset of a vehicle ahead reflect its actual speed, distance, and relative position.
Exp1-ESPS:	Measured speed and yaw rate of the ACC vehicle reflect its actual course.
Exp1-CAN:	Speed, distance, and lateral offset of vehicles ahead as well as speed and yaw rate of ACC vehicle transmitted by CAN correspond to the speed, distance, and lateral offset measured by the long range radar and the speed and yaw rate measured by the ESP sensors.
SOF1-ACC:	Determine course of the ACC vehicle based on the measured speed and yaw rate of the ACC vehicle. Calculate relative position of vehicle ahead based on determined course and lateral offset of vehicle ahead. Calculate engine- or braking-torque-to-be.
Exp2-CAN:	Transmitted engine- and braking-torque-to-be correspond to engineand braking-torque-to-be provided by ACC.
Exp1-ESP:	Decelerate instruction reflects braking-torque-to-be.
Exp1-EMS:	Accelerate instruction reflects engine-torque-to-be.
Exp1-ACCV:	Speed of ACC vehicle is adapted according to decelerate instruction.
Exp2-ACCV:	Speed of ACC vehicle is adapted according to accelerate instruction.

Fig. 7: Problem diagram for the ACC and environmental assumptions

although its input does not require deceleration (F ACCV1). This event covers exactly the situation when the c variable does not result in the d variable, i.e. negation of an assumption of type Exp-CD, here Exp1-ACCV. The ACC vehicle shall not decelerate but it does. The controlled domain ACC vehicle does not behave as expected. This situation may be seldom but nevertheless we consider it as a possible cause of the hazardous event. In Fault Tree 1, this event was not considered. The other possible cause of the root node is that the ACC vehicle decelerates due to the input it receives (FA ACCV DEC). This event is then further decomposed. The middle of the tree resembles Fault Tree 1. Yet, at the bottom of the tree, we find again differences, since we consider the possibility that the domain hypotheses DH1-ACCV and DH1-VA are wrong/invalid. There may be situations in which the course estimated by the ESP sensors does not correspond to the actual lane of the ACC vehicle (F ACCV2). In a curve, for example, the estimation of the own lane and the lane of vehicles ahead may be difficult. For example, if both cars are driving close to the road marking, the ACC may assume (based on the determined course of the ACC vehicle and the calculated relative position of the vehicle ahead) that they are driving on the same lane although they are actually on two different lanes. So, there may be situations in which the relative position of vehicles ahead does not reflect their lane. This is covered by event F LRR. Fault Tree 2 results in an overall failure rate of: $2.02 \cdot 10^5 h^{-1}$.

Comparison of the results. The comparison of the two fault trees reveals that, of course, the consideration of assumptions in Fault Tree 2 results in a higher value for the overall failure rate since more risks are considered than in Fault Tree 1. Yet, they are the assumptions that we actually make. Therefore, the overall failure rate of Fault Tree 2 is more realistic.

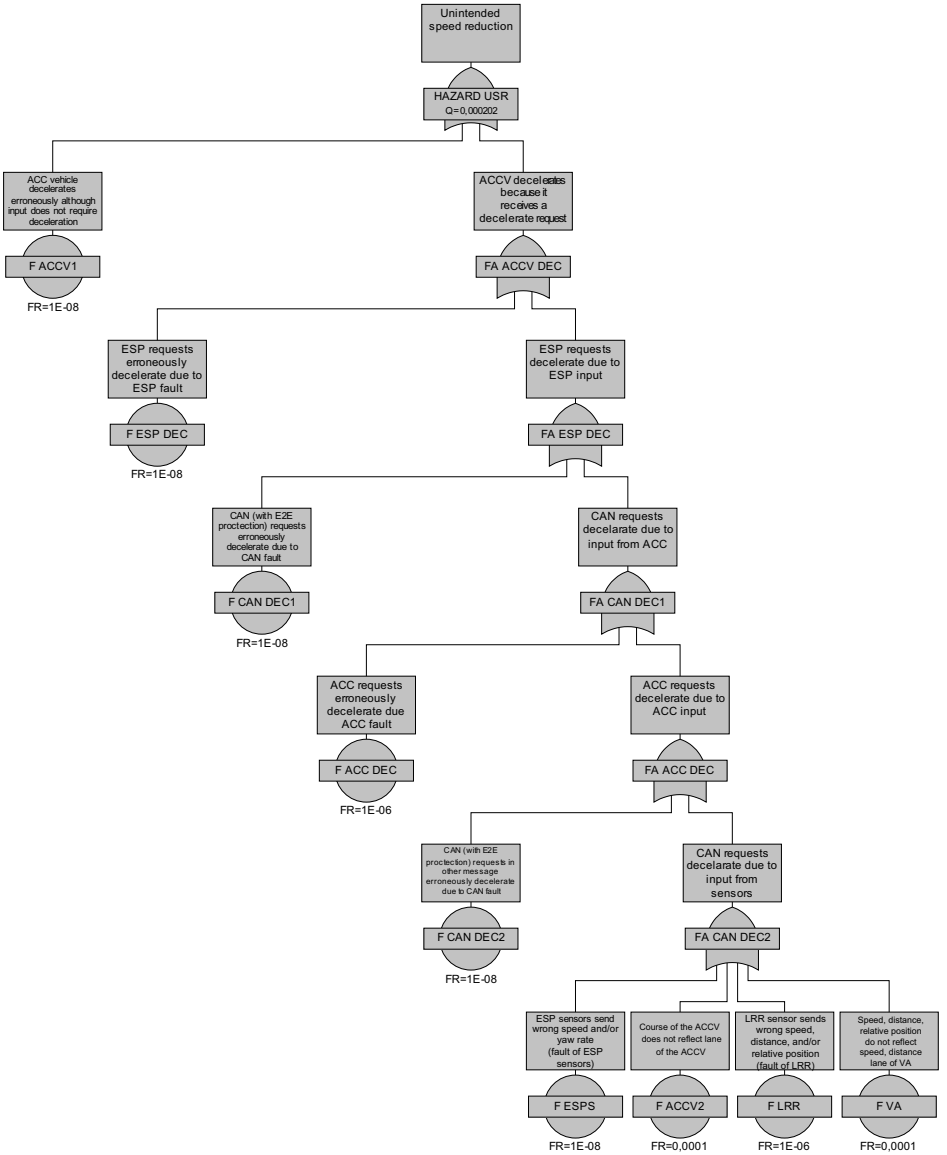


Fig. 8: Fault Tree 2 – Result of Our Method

Furthermore, it is more complete since we do not neglect failure events that may occur due to wrong/invalid assumptions (Exp1-ACCV, DH1-ACCV, DH1-VA).

6 Related Work

A method that is related to our work is the KAOS method described by van Lamsweerde [La09]. He assumes Jackson's model of the world and the machine and suggests a goal-oriented method. He also calls for documenting environmental assumptions (expectations and domain hypotheses) and for checking whether they are too strong and too risky. However, he focuses on the classical four variables and thus does not provide support in making the 'real' environmental assumptions explicit.

Another work that is closely related to ours is the work of Tun et al. [Tu15]. They suggest a method for identifying and classifying environmental assumptions whose violation is known from experience to have prevented the requirements from being satisfied. The idea is to reuse this knowledge of past failures to prevent failures in the future. The approach is quite interesting because it also considers possible mismatches between assumptions the software makes about the environment and the actual environmental reality. Yet, the method is reactive, i.e. failures must have been occurred in the past to be considered in the future. In contrast to that, our method is proactive. We provide support in making the environmental assumptions explicit and in considering the possibility for their violation already during safety analysis, i.e. before failures actually occur. The failures can then be prevented, for example, by changing the system design (e.g. adding sensors) and thus weakening assumptions.

Another related work is HAZOP (Hazard and Operability Studies) [IE01]. HAZOP is a technique for examining a system in order to identify potential hazards in the system as well as potential operability problems with the system. A key characteristic of HAZOP is that it provides a core set of guide words which are intended to stimulate the imagination of the team (performing the HAZOP study) in a systematic way to identify hazards and operability problems. We use these HAZOP guide words in our method (Step 3: Perform HARA) as well.

Finally, STPA (Systems-Theoretic Process Analysis) [Le11] is a further important related work. While fault tree analysis is based on reliability theory, STPA is based on systems thinking. This means, the cause of an accident/hazardous event is not understood as a series of failure events but as the result of a lack of constraints imposed on the development, design, and operation of the system. Safety is therefore viewed as a control problem, i.e. hazardous events occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not adequately handled. Preventing hazardous events requires thus designing a control structure that enforces the necessary constraints on system development and operation. STPA is an interesting method and, since it is based on systems thinking, is closer related to our Six-Variable Model than fault tree analysis. Yet, fault tree analysis is a well-established technique that is frequently used in industry, especially in the automotive domain. Therefore, our aim was to extend fault tree analysis. Yet, we consider extending STPA with our Six-Variable Model in future work.

7 Conclusion and Future Work

In this paper, we presented an ISO-26262-compliant method for performing hazard analysis and fault tree analysis. The main contribution of our method is that it supports the identification of failure events beyond the system border. More precisely, it supports developers in considering the risk associated with possibly too strong environmental assumptions in the hazard analysis as well as the fault tree analysis. We think that it is important to consider such risks because there are numerous real accidents that were caused by such assumptions turning out to be invalid in certain operational situations while the system behaved as specified. Too strong assumptions can be weakened or abandoned by changing the system design. Therefore, they should be detected early on in the development process. Our method supports that. In future work, we plan to extend STPA (Systems-Theoretic Process Analysis) with our Six-Variable Model and to compare the resulting method with the one described in this paper with regard to the applicability, usability, etc.

References

[Be13] Beckers, K.; Heisel, M.; Frese, T.; Hatebur, D.: A Structured and Model-based Hazard Analysis and Risk Assessment Method for Automotive Systems. In: Proc. of ISSRE 2013. pp. 238–247, 2013.

[IE01] IEC 61882 Hazard and Operability Studies – Application Guide, 2001.

[IS11] ISO 26262 Road Vehicles – Functional Safety, 2011.

[Ja01] Jackson, M.: Problem Frames – Analysing and Structuring Software Development Problems. Addison-Wesley, 2001.

[La09] Lamsweerde, A. Van: Requirements Engineering – From System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

[Le11] Leveson, N.: Engineering a Safer World – Systems Thinking Applied to Safety. The MIT Press, 2011.

[PM95] Parnas, D. L.; Madey, J.: Functional Documents for Computer Systems. Science of Computer Programming, 25(1):41–61, 1995.

[Ro03] Robert Bosch GmbH: ACC Adaptive Cruise Control - The Bosch Yellow Jackets. Edition 2003 edition, 2003.

[RS15] Ruijters, E.; Stoelinga, M.: Fault Tree Analysis: A Survey of the State-of-the-Art in Modeling, Analysis and Tools. Computer Science Review, 15(1):29–62, 2015.

[Tu15] Tun, T.; Lutz, R.; Nakayama, B.; Yu, Y.; Mathur, D.; Nuseibeh, B.: The Role of Environmental Assumptions in Failures of DNA Nanosystems. In: Proc. of COUFLESS 2015. pp. 27–33, 2015.

[UBMH16] Ulfat-Bunyadi, N.; Meis, R.; Heisel, M.: The Six-Variable Model – Context Modelling Enabling Systematic Reuse of Control Software. In: Proc. of ICSOFT-PT 2016. pp. 15–26, 2016.

[ZJ97] Zave, P.; Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, 6(1):1–30, 1997.