CALCUL DES METRIQUES D'ARCHITECTURE DE L'ISO 26262 A PARTIR D'ARBRES DE DEFAILLANCES

CALCULATION OF ISO 26262 ARCHITECTURAL METRICS FROM FAULT TREES

Abraham CHERFI and Michel LEEMAN
VALEO, GEEDS
2 rue André Boulle, 94046 Créteil Cedex
firstname.lastname@valeo.com

Antoine RAUZY
Chaire Blériot-Fabre, Ecole Centrale de Paris
Grande Voie des Vignes, 92295 Châtenay-Malabry
Antoine.Rauzy@ecp.fr

Résumé

La complexité et la criticité des systèmes électroniques embarqués automobiles est en augmentation constante. Un nouveau standard concernant la sûreté de fonctionnement automobile (ISO 26262) permet d'établir un cadre et de définir des exigences sur les systèmes concernés afin de garantir leur sécurité. Pour cela, L'ISO 26262 définie, entre autre, deux métriques architecturales (*Single Point Fault Metric* et *Latent Fault Metric*) permettant d'évaluer la robustesse d'un composant par rapport à un événement redouté en utilisant des AMDEC quantitatives. L'objet de cet article est de proposer une méthode alternative permettant le calcul de ces métriques d'architectures à partir d'arbres de défaillances. Nous présenterons donc une méthode basée sur des arbres de défaillances annotés permettant, entre autre, de déterminer la latence des fautes qui peuvent atteindre nos composants.

Summary

Cars embed a steadily increasing number of Electric and Electronic Systems. The ISO 26262 standard discusses at length the requirements that these systems must obey in order to guaranty their functional safety. One of the means at hand to evaluate a system's robustness with respect to a dangerous unwanted event is to evaluate two architectural metrics defined in the ISO26262. The aim of this article is to present a method to calculate these architectural metrics (*Single Point Fault Metric*, *Latent fault Metric*) from Fault Tree models. We propose a tag based approach that aims at documenting Fault Trees in order to identify fault latencies. This approach makes it possible to improve the productivity of the safety process by getting rid of the quantitative part of the Failure Mode, Effect and Diagnostic Analyses (FMEDA).

Introduction

Cars embed a steadily increasing number of Electric and Electronic Systems. In order to guaranty their Functional Safety, the ISO 26262 standard was published in November 2011[1, 2]. This standard defines a number of constraints, rules and requirements that the development of Automotive Electric and Electronic Systems must obey. It defines also three quantitative metrics to evaluate the safety of architectures with respect to random hardware failures: one classical probabilistic metric, and two architectural metrics that are specific to the ISO 26262 Standard.

These two architectural metrics can be calculated using quantitative FMEAs as presented in annex F of ISO 26262 - part 5 [1]. As the probabilistic metric is anyway calculated from fault tree models, it is worth to investigate the possibility to compute the architectural metrics from such a model as well. To our knowledge, only very few investigations have been done in this direction. We can quote basically only one communication in CTI Conference 2012 [4]. There is however currently works in the SAFE Project for the calculation of architectural metrics from models [5].

In this article, we propose a tag based approach that aims at documenting Fault Trees in order to identify fault latencies. The metrics can then be calculated from the minimal cut sets generated from the Fault Tree. This process requires the design of a tagging methodology as well as of an algorithm to calculate the metrics from the minimal cut sets. This is very purpose of the present communication.

The remainder of this article is organized as follows: First, we introduce examples of common automotive systems and their specificities. Then, we present the ISO 26262 architectural metrics. Then, we introduce a fault classification approach for faults latency. Finally, we describe our fault tree patterns and our tag based fault tree calculation approach.

Examples of Automotive Systems with Safety Mechanisms

In order to explain our needs and constraints, we present in this section a typical example of automotive systems embedding safety mechanisms.

Vehicle Management Unit for Inversion

In an electric vehicle, the Vehicle Management Unit (VMU) is responsible for commanding the electric motor inverter, among other functions. A VMU consists typically in a microcontroller which, given certain inputs (gas and brake pedal positions), sends a torque set-point to the inverter that in turn commands the electric motor (traction and regenerative braking), as illustrated in Figure 1.

Such a VMU is a critical function: if the microcontroller gets stuck in a loop and continuously sends a command higher (or lower) than expected, it could lead to unintended vehicle acceleration or braking.

To prevent such hazards, a watchdog is added: it is in charge of bringing the system to a safe state when the microcontroller is detected to be stuck. The watchdog is an electronic component that is used to detect and recover from microcontroller malfunctions. The microcontroller refreshes regularly the watchdog in order to prevent him from timing out. If it gets stuck in a loop, the watchdog cannot be reset, so the watchdog times out and sends a reboot order to the microcontroller.

Such a watchdog is a first order safety mechanism based on error detection.

As a physical component, the watchdog may fail (although the reliability of the watchdog is much higher than the one of the microcontroller). Also, the watchdog is able to detect only certain kinds of errors of the microcontroller: typically, it is not able to detect data memory corruption problems.

In order to ensure that the watchdog is working, the microcontroller tests the watchdog at each vehicle start. The role of this second order mechanism is to warn the driver in case of a problem with the watchdog. It may itself fail and is itself not able to catch all of the problems of the watchdog.

As the torque calculation function and the second order safety mechanism function are never executed in parallel, their failures are considered as independent (and are independent from watchdog failures).

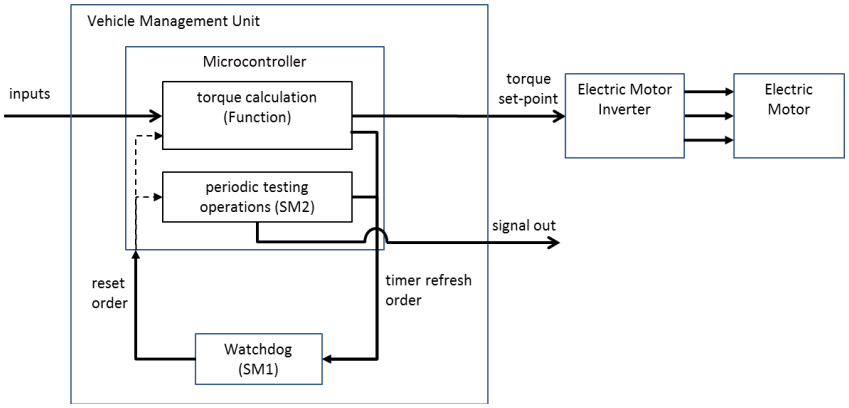

Figure 1. Simplified functional representation of the Vehicle Management Unit for Inversion

The above example is representative of safety mechanisms based on error detection as embedded for instance in Torque Calculation functions, electric braking, several types of microcontrollers protected with watchdogs and more generally command-control systems.

Complements and Generalizations

In addition to the detection based safety mechanisms, there are also safety mechanisms based on inhibition, as embedded for instance in Electric Steering Column Lock, Automatic Doors opening systems and more generally all systems that must be inhibited when the speed of the vehicle gets above a given threshold.

In fact, the majority of automotive first order safety mechanisms can be actually categorized in either of the two categories presented above:

- Most of them are based on error detection. The idea is to switch the system into a safe state when an error is detected. These safety mechanisms are usually made of two elements: the detection device and the actuation device. In addition to the VMU for inversion (ASIL C), in this category we have as examples :
    - Steering angle sensors (ASIL D): In order to avoid an unwanted regulation of the electronic stability program, a safety mechanism is in charge of checking the validity of the data sent to it from the steering angle sensors. When it detects wrong information coming from the sensors, it disables the electronic stability program assistance.
    - Wipers activation unit (ASIL A/B): The inability to set on wipers in case of strong rain could lead to visibility loss problems. Therefore, we implement a safety mechanism that is in charge of verifying that the wipers can be commanded, and if not, it forces the wiper motor activation.
- Some of them inhibit the system they protect when the vehicle is in a state where the failure of the system is potentially dangerous, in this category we have the following examples :
    - Electric steering column lock (ASIL D): In order to avoid the electric steering column lock activation when the vehicle is moving, we implement a safety mechanism that is in charge of switching off the electric power supply of this component when a certain speed is reached (usually 4km/h). Indeed, a spurious activation of this component while the vehicle is moving at common speed could lead to the loss of the vehicle control.
    - Electric driver seat control: (ASIL B): In this case too, the supply of the electric driver seat control is inhibited by a safety mechanism when a certain speed (10km/h) is reached, in order to avoid spurious change of the driver seat position. Without this safety mechanism, this could easily cause an accident by making the braking pedal unreachable or by suddenly pushing the driver on the steering wheel.

In this paper however will only focus on the safety mechanism based on detection, as they are the most common and the trickiest to handle and represent.

For this kind of safety mechanisms, as a failure of the first order safety mechanism failure has in general no direct influence on the system itself, it can hardly be perceived by the driver. A second order safety mechanism is thus often added in order to check periodically the availability of the first one, typically when the engine is turned on before the vehicle starts to move. The role of such a second order mechanism is only to warn the driver.

The Example illustrated in Figure 1 will also serve in the next sections to present the notions that are going to be introduced there.

ISO 26262 Architectural Metrics

An automotive system is composed by hardware parts; each of them has a specific function and can be subject to faults and failures which could potentially lead to an unwanted event. In order to prevent the propagation of these faults, an automotive system also implements safety mechanisms; most of these devices purpose is to detect hardware parts failures and prevent their propagation with a certain diagnostic coverage percentage, and as they are considered as hardware parts, they can also be subject to failures.

ISO 26262 defines two architectural metrics (Single Point Fault Metric and Latent Fault Metric) to estimate the proportion in a component of certain types of fault with regard to all the faults that can attain that component causing a certain unwanted event. In order to present these two metrics, it is necessary to present the different fault types that can attain the automotive systems. Table 1 presents the different hardware failure that are considered and their corresponding failure rates.

Table 1. ISO 26262 definition of fault types

Fault	Description	Failure Rate
Basic faults	Each fault that can attain a hardware part in our systems can be considered as a basic fault either it causes the system failure or not. It can be considered as the sum of all the failure modes that an assessed hardware part can be subject to.	λ
Singles Point Fault	A fault can be considered as a Single Point Fault if its occurrence directly implies the occurrence of an unwanted event in the system.	λ_{SPF}
Residual Faults	It is a fault that directly causes an unwanted event, even if the responsible hardware block failure is covered by a safety mechanism probably due an imperfect diagnostic coverage	λ_{RF}
Multiple Point Faults	These are fault that cannot directly lead to the occurrence of a hardware event, but instead, their combination can do it.	λ_{MPF}
Multiple Point Fault (Latent)	These are the Multiple Point Faults that cannot be detected nor perceived, so they stay latent in the system until the apparition of other multiple fault that can lead to an unwanted event.	$\lambda_{MPF,Latent}$
Multiple Point Fault (Perceived or detected)	These are the Multiple Point Faults that can be perceived by the driver, either by the help of a safety mechanism signalization or by performance degradation.	$\lambda_{MPF,Perceived\ or\ Detected} = \lambda_{MPF} - \lambda_{MPF,Latent}$
Safe Fault	These are the faults that cannot lead to the Assessed unwanted event. In practice if a Multiple Point fault requires several other ones for the occurrence of an Unwanted Event than we consider it as a Safe fault	$\lambda_S = \lambda - \lambda_{RF} - \lambda_{SPF} - \lambda_{MPF}$

Single Point Fault Metric (SPFM)

The Single Point Fault Metric (SPFM) represents the proportions of random hardware faults that do not directly lead to the occurrence of an unwanted event, thus, it represents the robustness of the item that is assessed to the single point and residual faults. It is defined in the ISO 2626 by the following formula:

$$1 - \frac{\sum_{SR,HW}(\lambda_{SPF} + \lambda_{RF})}{\sum_{SR,HW}\lambda} = \frac{\sum_{SR,HW}(\lambda_{MPF} + \lambda_{S})}{\sum_{SR,HW}\lambda}$$

As we can see, for the calculation of this metric, we need to consider and be able to identify the following:

- The Single point faults and residual faults failure rates for the numerator calculation $(\lambda_{SPF} + \lambda_{RF})$,
- The sum of all the safety related failure rates (λ) for the denominator.

Latent Fault Metric (LFM)

The Latent Fault Metric (LFM) reflects the robustness of a system regarding to the latent fault that leads to a specific unwanted event. It represents the proportion of multiple faults that do not remain unnoticed in the system and that could lead to the occurrence of a hazard. It is defined in the ISO 26262 by the following formula:

$$1 - \frac{\sum_{SR,HW} (\lambda_{MPF,latent})}{\sum_{SR,HW} (\lambda - \lambda_{SPF} - \lambda_{RF})} = \frac{\sum_{SR,HW} (\lambda_{MPF,perceived\ or\ detected} + \lambda_S)}{\sum_{SR,HW} (\lambda - \lambda_{SPF} - \lambda_{RF})}$$

As we can see, for the calculation of this metric, we need to consider and be able to identify the following:

- The sum of all Latent Multiple Point Faults failure$(\lambda_{MPF,Latent})$ rates for the numerator calculation,
- The sum of all the Safety Related Faults, the Single Point Faults and the residual faults failure rates $(\lambda - \lambda_{SPF} - \lambda_{RF})$, for the denominator.

In reference [3], L'Hostis proposed a simplified method for the calculation of these metrics. This method makes it possible to get approximated values without having to dive down into hardware parts, hence saving time of the analyst. The obtained values are always pessimistic and the method is thus conservative. In fact, the approximation comes from that L'Hostis considers only what he calls Safety Related Basic Event failure rates. These failure rates correspond to the different failure modes of hardware block which contribute to the occurrence of a basic event, which itself can lead to a failure of the whole system.

Since the failure rates that are contained into the Fault Tree model are exactly the same as those considered L'Hostis's, the results of the two methods will be the same. The benefit of our approach is that all of the relevant metrics can be calculated from the unique fault tree model and ad-hoc separated study is unnecessary.

In the following section, we will present a process to categorize the basic events.

Fault Classification

In order to compute the two architectural metrics, it is necessary to distinguish different categories of basic events. The chart presented Figure 2, makes it possible for the analyst to split Basic Events into 3 categories:

- Single Point Faults and Residual Faults that show up in the numerator of the SPFM definition.
- Latent Faults that show up in the numerator of the LFM definition.
- And finally, the remaining faults that are safety related, but that show up neither in the SPFM nor the LFM numerators.

This is a practical reinterpretation of the fault diagram in ISO 26262 Part 5 Annex B [1].

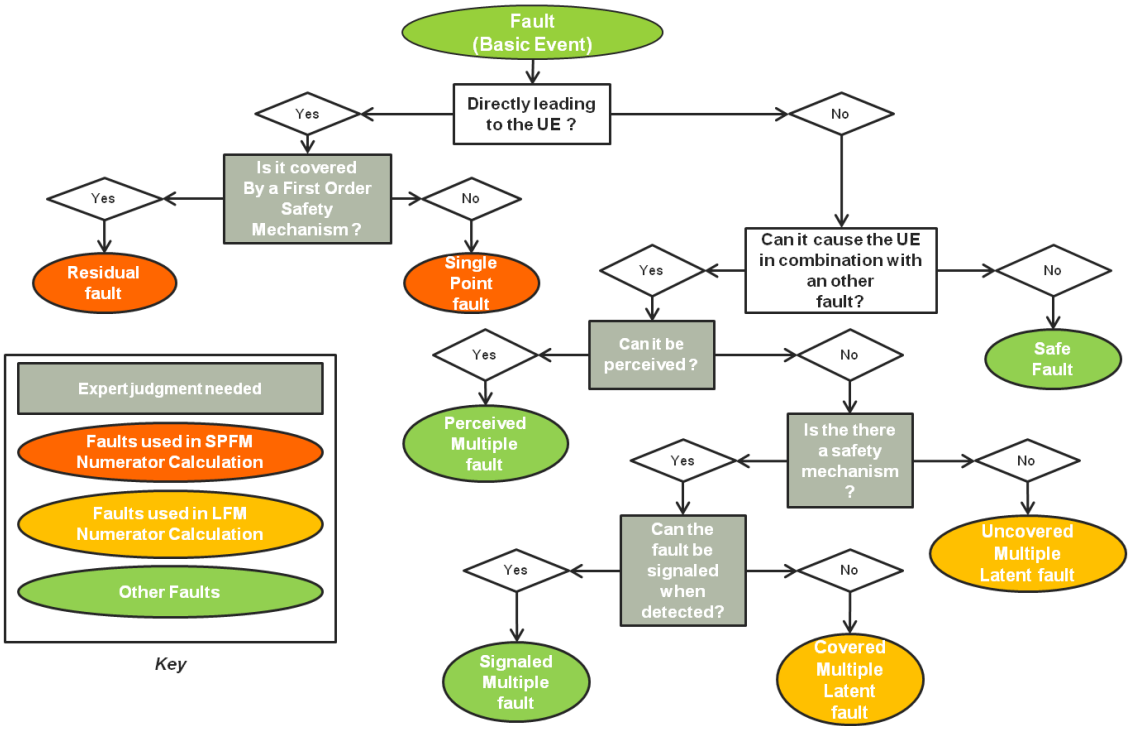

Figure 2 – Rearranged fault classification diagram

In the chart Figure 2 we clearly see that there are 4 important data that allow us to determine the category of fault (Basic Event):

- If the fault directly leads to the component failure, it is taken into account in the SPFM numerator $(\lambda_{SPF} + \lambda_{RF})$.
- If it can lead to the component failure in combination with another fault (λ_{MPF}) :
  - If it can be perceived by the driver (for example, performance degradation) it's a Perceived Multiple Point Fault $(\lambda_{MPF,Perceived})$.
  - Else, If the fault can lead to the component failure and is covered and detected by a safety mechanism :
    - If the safety mechanism can alert the driver, than it is considered as a detected fault$(\lambda_{MPF,Detected})$.
    - Else, it is a latent fault $(\lambda_{MPF,Latent})$.
  - If the fault does not lead at all to the assessed unwanted event, it's directly categorized as safe fault (λ_S).

On one hand, as seen in Figure 2, the basic events failure rates that are taken into account in the SPFM numerator can be easily extracted from the first order cut sets of a fault tree. On the other hand, there is no way to efficiently identify the latent multiple points from the detected ones using a simple fault tree for the LFM calculation. This is why we will define tags based on given fault tree patterns in order to be able to categorize faults and failures latency and thus, computing the second architectural metric.

A Tag Based Approach

In this section we present a method based on tagged fault trees for the calculation of the simplified ISO 26262 architectural metrics, however, as there is no general representation of second order safety mechanism in ISO 26262 document, we first present different fault trees patterns for this purpose

Fault Tree Patterns for the representation of first and second order safety mechanisms

In this section we present 3 possible fault tree models for representation of the failure of a block and its two safety mechanisms. Each of these models features a different way of implementing the second order safety mechanism: Indeed, the fault tree representation of a function failure with a first order safety mechanism (Figure 3) is documented in ISO 26262 part 10 Figure B.4 [2]. However, there are no generalized fault trees for the representation of the second orders safety mechanisms.

It should be noted that the accuracy of each of these patterns has been tested by comparing the obtained failure probability with Markov models [6] built for the representation of safety mechanism behavior.

As the fault tree representation of a function failure with a first order safety mechanism (Figure 3) is documented in ISO 26262 part 10 Figure B.4 [2], the main difference between each of these models is in the representation of the second order safety mechanism.

The models represented in this section are all implementable using the OpenPSA format [7].

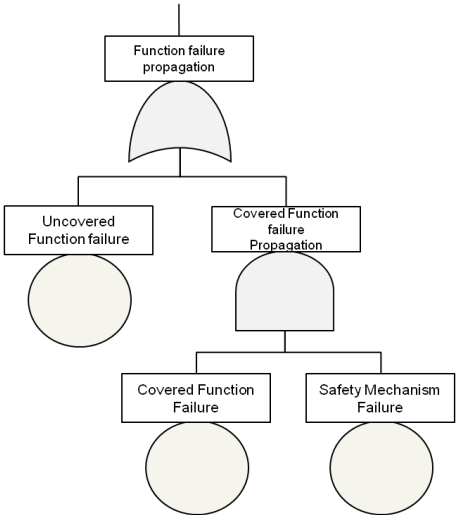

Figure 3. ISO 26262 fault tree representation of a function failure with first order Safety Mechanism

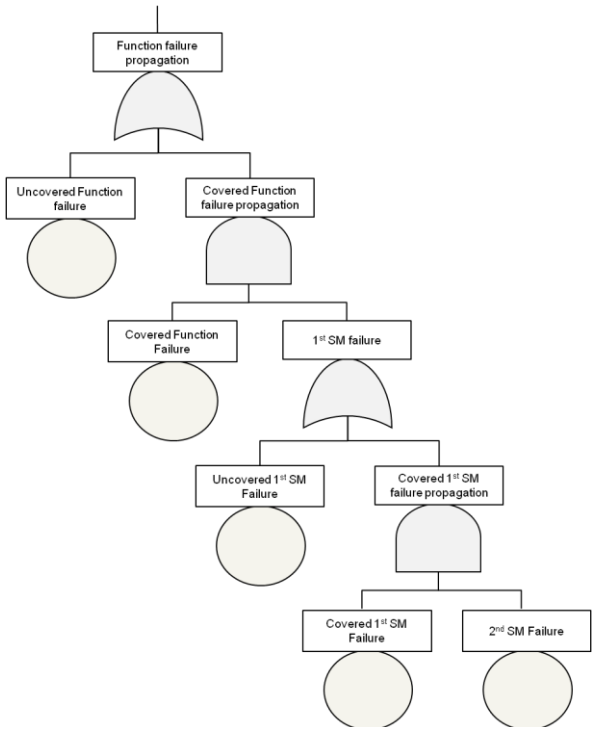

Figure 4. Fault tree pattern with a Second Mechanism represent as a classic Safety Mechanism

1.1.1 FT Model with Classic SM Representation for SM2

In this Model(Figure 4), we consider and represent the second order safety mechanism SM2 as if it was a first order safety mechanism applied on SM1. So, we represent each of them using the classic OR/AND pattern (presented in ISO26262 part 10 [2]).

We consider five basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The covered part of first order safety mechanism failure, following an exponential law with the parameter DC2 * λ_{SM1}
- The uncovered part of the first order safety mechanism failure, following an exponential law with the parameter [(1-DC2) * λ_{SM1}]
- The second order safety mechanism failure, following an exponential law with the parameter λ_{SM2}

It shall be noted that this model can't consider the test interval and the time before going to maintenance. However, as long as the SM2 don't fail, the failure of the covered part of SM1 cannot be propagated.

1.1.2 FT Model with Maintenance

In this model (Figure 5), we consider that when the second order safety mechanism SM2 detects the first order safety mechanism failure, it leads to maintenance (and the repair) with a periodic maintenance rate. Thus, this model does not directly consider the possibility of the second mechanism failure but rather focuses on its action.

This fault tree model is based on the generalization and the adaptation of examples from ISO 26262 part 10. Amongst others, the figures B.15 and B.18 [2] are two examples of the use of this model.
However, as a test do not guaranties the maintenance and the repair of the vehicle, we adapted this model to consider the use of maintenance frequencies instead of tests frequencies as shown in those examples.
So, we consider four basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The covered part of first order safety mechanism failure, following an exponential law with the parameter DC2 * λ_{SM1}
- The uncovered part of the first order safety mechanism failure, following a GLM distribution with failure rate [(1-DC2) * λ_{SM1}], a reparation rate μ_V and an failure on demand probability (fixed to 0)

This model can be seen as a generalization of the second safety mechanism representation in ISO 26262 part 10 Fault tree examples.

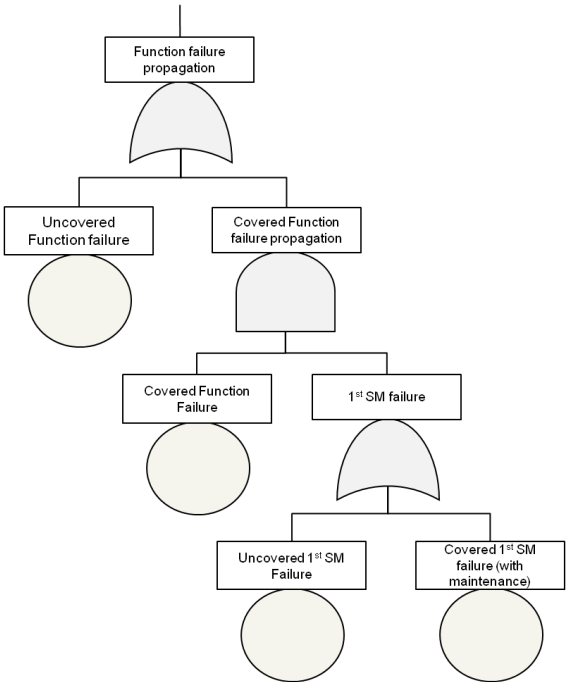

Figure 5. Fault tree pattern that takes into account the maintenance action of the 2nd order Safety mechanism

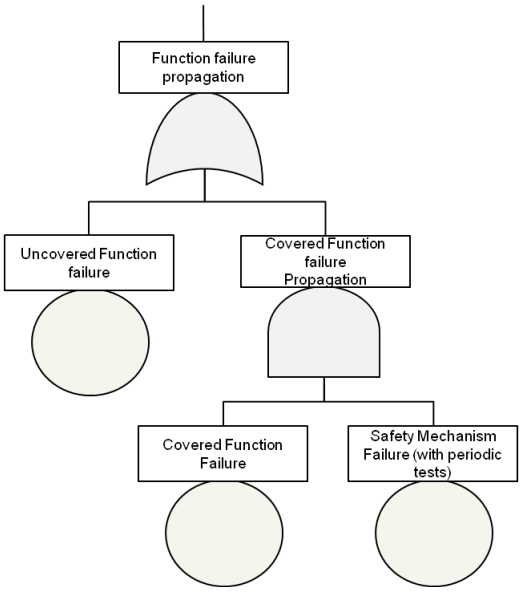

Figure 6. Fault tree pattern for the representation of the second order safety mechanism periodical testing behavior

1.1.3 FT Model with Periodic Tests

Like the previous model, this one (Figure 6) focuses on the representation of the second order safety mechanism (SM2) action. In this model, in addition to maintenances interval, we also consider SM2 tests periodicity. This is made possible by using the periodic test law defined in OpenPSA [7].

So, we consider three basic events:

- The covered part of the block failure, following an exponential law with the parameter DC1 * λ_{HB}
- The uncovered part of the block failure, following an exponential law with the parameter [(1-DC1) * λ_{HB}]
- The first order safety mechanism failure, following a periodic test law with the following parameters :
  - The failure rate of SM1 when working: λ_{SM1}
  - The failure rate of SM1 when being tested: λ_{SM1}
  - The repair rate of SM1 (when detected): μ_V
  - The Delay between two consecutive tests : $1/\delta_V$
  - The Delay before the first test: $1/\delta_V$
  - The probability of failure due to the test: 0 (not considered)
  - The duration of the tests : 0 (as the tests are not done during the vehicle service)
  - The availability of the component during the test : 1
  - The probability that detects a failure (if any) : DC2
  - The probability that the component is badly restarted after repair : 0 (as the maintenance is out of the scope of ISO26262)

According to the tests that we made [6], of all the presented models, this one offers the best approximation of the failure probability of an automotive system, even when the diagnostic coverage of the first safety mechanism is high.

Custom Fault Tree patterns with Coverage Gate

As the previously presented patterns shows big differences in their constructions, we propose in this subpart a custom fault tree like pattern for the representation of first and second order safety mechanisms.
This pattern uses a custom type of gate that we will name “coverage gate”. And which can be implemented at least using OpenPSA Format [7]. The use of this pattern and gate has two main advantages:

- It contains all the required data to be easily parsed and translated into any of the previously described patterns,
- It is compact, making it easy to handle.

1.1.4 Coverage Gate Definition

We define the coverage gate as an asymmetric gate that has one output, two inputs and an internal parameter. The first input is always the covered basic event, and the second input is always a safety mechanism. The internal parameter is labeled as “DC” and contains the value of the diagnostic coverage of the safety mechanism toward the basic event.

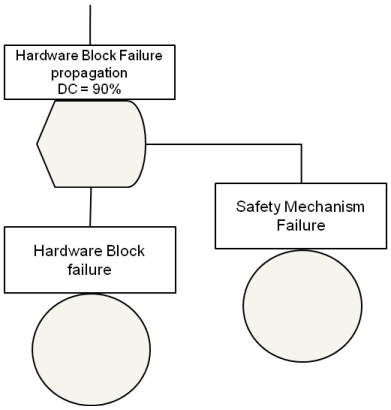

Figure 7. Graphical representation of a coverage gate

The covered basic event must have a failure rate. The safety mechanism must also have a failure rate, and tests and maintenances intervals if it is categorized as a second order Safety Mechanism. With these data filled, the generation of any of the previous patterns is possible.

1.1.5 Minimal cut-sets extraction

For the purpose of our tag based approach, we built a script that computes the minimal cut-sets of fault trees using this gate by taking into account the diagnostic coverages of the safety mechanisms. The cut sets generated from this pattern are the same as the one that would be generated using the fault tree pattern with classical representation of SM2 (Figure 4).
For example, for the fault tree branch in Figure 7, the cut sets would be:

- A first order cut set representing the uncovered part of “HB Failure” (with 10% of the original failure rate).
- A second order cut set representing the combination of the covered part of “HB Failure” (with 90% of the original failure rate) and “SM Failure” with its full failure rate.

1.2 Tag based approach for the architectural metrics calculation

In order to compute the architectural metrics from fault trees using the previously presented custom pattern, we propose a tag based approach to complete the data that can intuitively be accessed in our patterned fault trees. The main purpose behind these tags is to be able to determine fault latency.

1.2.1 Tag presentation

Based on the fault classification presented in section 4 (Figure 2), we managed to define three tags:

- The first one is used to indicate if the occurrence of a basic event can naturally be perceived, we associate to it a value between 0 and 100, which represent the percentage of the perception probability.
  - We represent it by the form: [Perceived,X], where X is a percentage between 0 and 100%.
  - When the minimal cut sets are extracted from a fault tree containing Coverage gates, this tag is inherited by both the covered and the incovered basic event generated from a tagged basic event.

- The second one is used to indicate if the basic event occurrence is covered by a safety mechanism that alerts when it detects its failure. This tag could be binary, however, we chose to associate a percentage to it in order to take into account the probability that the driver misses or ignores the signal/alert.
  - We represent it by the form: [Signaled,X], where X is a percentage between 0 and 100%.
  - When the minimal cut sets are extracted from a fault tree containing Coverage gates, this tag is inherited only by the covered basic event generated from a tagged basic event.

- If the basic event occurrence correspond to a 2nd order safety mechanism failure, it is necessary to tag it in order to signify that it considered as a safe fault. Indeed, as the second order safety mechanism have no direct impact on failure propagations, there are considered as safe faults in ISO 26262.
  - We use the tag [SecondOrder]. This tag can be placed automatically in our custom pattern, as the second order safety mechanisms are the only ones that have a test interval.

1.2.2 Tag based algorithm for architectural metric calculation with fault tree

In this section, we will present a pseudo-algorithm based on the tags previously presented and the fault trees structure in order to compute the architectural metrics calculation.
For the safety mechanisms presentation, we use the previously defined fault tree pattern.

```
Ignored = {};     Perceived = {};
Single = {};      Signaled = {};
Latent = {};      Safe = {};


Extract all the cut-sets from the fault tree with their tags;
for each cut-set with an order > N
      for each basic event BE in the cut-set
            Ignored = Ignored + {BE};
      end ;
end ;

for each cut-set with an order = 1
      Single := Single + {Be};
end ;

for any other cut-set order
      for each basic event Be in each cut-set
            if the basic event is tagged by [SecondOrder]
                  Safe :=  Safe + {Be};
            elseif the basic event has the tag [Perceived,X] (where X is a number)
                  BE.coefficient := X from [Perceived,X];
                  Perceived := Perceived + {Be};
            elseif the basic event has the tag [Signaled,X] (where X is an number)
                  BE.coefficient := X from [Signaled,X];
                  Signaled := Signaled + {Be};
            else
                  Latent := Latent + {Be} ;
            end ;
      end ;
end ;
LambdaIgnored = Sum of Be.Lambda in Ignored;
LambdaSingle = Sum of Be.Lambda in Single;
LambdaLatent = (Sum of Be.Lambda in Latent) + (Sum of Be.Lambda*(1-Be.coefficient) in Perceived and Signaled);
LambdaSafe = (Sum of Be.Lambda on Safe) + (Sum of Be.Lambda*Be.X on Perceived and Signaled);
```

As we can see, the first thing to do is to extract all the cut set in the fault tree. Then, based on the tags, we calculate the sum of the failure rates corresponding to each category of fault.
By using this algorithm result, we're able to compute the architectural metric parameters assuming the following equivalences:

- $\sum_{SR;HW}(\lambda_{SPF} + \lambda_{RF}) = LambdaSingle$; Because the first order cut-sets represent the fault directly leading to the unwanted event.
- $\sum_{SR;HW}(\lambda_{MPF,Latent}) = LambdaLatent$; Because this represent the part of the basic event which are are not perceived nor alerted
- $\sum_{SR;HW}(\lambda_{safe}) = LambdaIgnored + LambdaSafe$; Because this represent the combination of basic event that are of a too high order to be considered or that are perceived or alerted.
- $\sum_{SR;HW}(\lambda_{SRBE}) = LambdaIgnored + LambdaSafe + LambdaLatent + LambdaSingle$; The sum of safety related basic event failure rates as defined in reference [3].

Thus, we have:

$$SPFM = 1 - \frac{LambdaSingle}{LambdaTotal} \quad , \quad LFM = 1 - \frac{LambdaLatent}{LambdaTotal - LambdaSingle}$$

Application examples

In this sub-section, we will present an example of the usage of the previously presented tag based approach. We will first build an example fault tree using "coverage gates" based on the Vehicle management unit presented in Figure 1: We consider that the electric motor receives a wrong three phase current if either the Electric Motor inverter or the Vehicle Management Unit is attained by a failure.

For the purpose of simplification, we will assume that the Electric Motor Inverter failure is a basic event. We than obtain the fault tree presented in Figure 8.

Also, we consider the PTU and the TCU as independent, because their two functions are never executed at the same time even if they are supported by the same hardware (as seen in Figure 1).

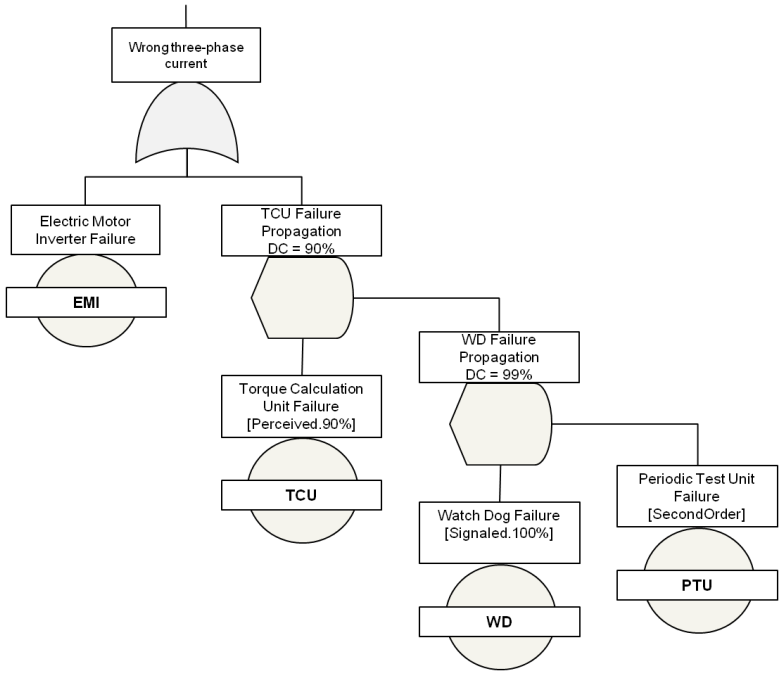

Figure 8. Fault tree simplified example for the representation of the generation of a wrong three-phase current for an electric motor

The basic events presented in this fault tree are all characterized by their failure rates (Table 2). The coverage of each safety mechanism toward the basic event it covers is present on the corresponding coverage gate (90% for the Watch Dog and 99% for the Periodic Test Unit).

Table 3. Basic events failure rates for the wrong three phase current generation fault tree

Basic event	Failure rate name	Failure rate value
Electric Inverter Unit failure	λ_{EIU}	$5e^{-8}$
Torque Calculation Unit failure	λ_{TCU}	$1e^{-6}$
Watch Dog failure	λ_{WD}	$1e^{-7}$
Periodic Test	λ_{PTU}	$5e^{-8}$

As the Periodic Test Unit is a second order safety mechanism, it should also be characterized by a test frequency and a maintenance frequency.

We than proceed to the minimal cut sets extraction as presented in Section 5.2.2. We obtain four cut sets composed with 5 basic events. Two cut sets of the first order:

- The first one is composed by the Electric Inverter Unit failure with a failure rate λ_{EIU},
- The second one is composed of the uncovered part of Torque calculation unit failure. Its failure rate is obtained by inverting the related DC : λ_{UTCU} = 10% λ_{TCU}.

We also obtain a second order cut set composed by:

- The covered part of the Torque calculation Unit, with a failure rate λ_{UTCU} = 90% λ_{TCU}.
- The uncovered part of the Watch Dog failure, with a failure rate λ_{UWD} = 1% λ_{WD}.

There is also a third order cut set composed by:

- The covered part of the Torque calculation Unit, with a failure rate λ_{UTCU} = 90% λ_{TCU}.
- The covered part of the Watch Dog failure, with a failure rate λ_{DWD} = 99% λ_{WD}.
- The Second order safety mechanism failure λ_{PTU}.

For each of the basic events that we obtain in this cut sets, we inherit the tags of their original event. Then we apply the previously presented algorithm. The obtained results are the following:

LambdaIgnored = 0

LambdaSingle = $\lambda_{UTCU} + \lambda_{EIU} = 5e^{-8} + 1e^{-7} = 1.5e^{-7}$

LambdaLatent = $\lambda_{UWD} + (1\text{-}90\%)\lambda_{DTCU} + (1\text{-}100\%)\lambda_{DWD} = 1e^{-9} + 10\%\ 9e^{-7} + 0 = 9.1e^{-8}$

Thus, we finally obtain:

$$\boldsymbol{SPFM = 1 - \frac{LambdaSingle}{LambdaTotal} = 87.5\%} \quad , \quad \boldsymbol{LFM = 1 - \frac{LambdaLatent}{LambdaTotal - LambdaSingle} = 88,75\%}$$

Other tests have been led on more consistent examples directly extracted from the ISO 26262, the obtained results were exactly the same as the ones obtained with approach proposed by L'Hostis [3]. Although, these are not the exact calculation of the metrics, it offers good pessimistic approximations.

Conclusion

In this article we presented a method which allows us the approximative calculation ISO26262 architectural metrics using tagged fault trees. This method represents a good alternative for the classic quantitative FMEA method proposed in the automotive safety standard. We actually see two axes that could allow making this method more efficient: We are currently investigating the previous defined tagged pattern generation from higher level models, making their usage more convenient and time saving. We also are investigating a way to combine the fault corresponding to the different unwanted event that could attain a system in order to allow the exact calculation of the sum of the safety related failure rates and thus, be able to compute the exact metrics values. Also, an alternative that would be worth investigating is to combine the fault trees with the electronic FMEA data.

References

[1] ISO: "ISO26262 *Functional Safety – Road Vehicle*", Part 5, 2011.

[2] ISO: "ISO26262 *Functional Safety – Road Vehicle*", Part 10, 2012.

[3] Architectural metrics calculation – an efficient approach, S. L'Hostis, SIA – Journée d'Etude Sécurité Fonctionnelle Electronique Automobile, November 2013.

[4] A practical Approach to Calculation of ISO 26262 Metrics for ASIL C/D Critical Systems, T. Stanyer, CTI Conference, 2012.

[5] Safe Project, "Safe Automotive soFtware architecture", http://www.safe-project.eu/.

[6] Modeling Automotive Safety Mechanisms: A Markovian Approach, A. Cherfi, A. Rauzy, M. Leeman, F. Meurville, Reliability Engineering and System Safety, Accepted in April 2014.

[7] Open-PSA Model Exchange format, version 2.0d, S. Epstein, A. Rauzy, 2008.