Towards Vehicle-Level Simulator Aided Failure Mode, Effect, and Diagnostic Analysis of Automotive Power Electronics Items

Towards Vehicle-Level Simulator Aided Failure Mode, Effect, and Diagnostic Analysis of Automotive Power Electronics Items

J. Sini, M. D'Auria, M. Violante
Control and Computer Engineering Department (DAUIN)
Politecnico di Torino
jacopo.sini@polito.it, marco.dauria@studenti.polito.it, massimo.violante@polito.it

Abstract—**The increasing demand for Electronic Control Units able to perform safety-relevant tasks leads the automotive industry to find novel verification methodologies, capable to decrease the time-to-market and, at the same time, to improve the quality of the assessment. The ISO26262:2018 automotive functional safety standard requires to follow a strict development process, compliant with its "safety lifecycle". It includes all the phases of the item life, from the concept to the decommissioning. The phase that places most difficulties about its objectivity and repeatability is the hardware/software integration verification since, usually, the software is in charge to mitigate the effects of some possible hardware failures. This paper proposes a novel technique, based on a simulation-based approach, to aid the designers during the Failure Mode, Effect, and Diagnostic Analysis (FMEDA). We consider a power electronics module, to be embedded into electric vehicles powertrains, as a challenging practical example. We performed some tests on it, considering a rear traction car with two independent electric motors, one per each wheel. This system, to allow the vehicle to curve, has to act like a differential gear. Hence, it has a strong safety impact on the driveability of the car. All the involved components have been simulated propagating their behaviours up to the entire vehicle. Due the strong coupling between item failures and vehicle dynamics, a structured way based on coupling fault injection with vehicle dynamic simulation is desirable.**

Keywords—. ***Circuit faults; Hardware; Software; Microcontrollers; Safety; Automotive electronics; Embedded systems; failure analysis; ISO 26262 standard; Reliability***

I. INTRODUCTION

Electronics and electrical (E/E) items design is becoming day by day more important in the design process of new cars. In particular, to develop electrically powered vehicles (EV), the design of high-reliable electric components is becoming a crucial point for automotive companies.

Since 2011 the ISO 26262 [1] standard prescribes the safety lifecycle that has to be followed during the development of those items that are in charge of safety-related functionalities. In 2018 the Standard has been upgraded to be applied to all road vehicles except for mopeds and aftermarket parts for vehicles designed to be operated by drivers with disabilities.

In this paper, we propose a novel methodology to improve the Failure Mode, Effect, and Diagnostic Analysis (FMEDA), by adopting a simulation-based approach. We propose to simulate the item, obtaining its possible misbehaviors by injecting the possible faults that can affect it. By propagating these misbehaviors to the vehicle-level though a vehicle dynamics simulator, it will become possible to perform the failure effect classification by keeping into account the forecasted effects on the dynamics and drivability of the vehicle. This can be particularly useful in those cases where the behavior of the item is highly coupled with the behavior of the whole vehicle.

II. BACKGROUND

To better understand the problem, it is convenient to briefly describe the ISO 26262 "safety lifecycle".
This Standard puts the safety aspects in the central position of the whole design process.
The ISO 26262 Part 5 [1], titled "Product development at the hardware level", describes how to verify the hardware designs in charge to provide safety-relevant functions of a vehicle. One of the instruments to compute these reliability metrics is the FMEDA. Safety-relevant effects of each vehicle function are analyzed keeping into account the whole lifecycle of an item, starting from the concept phase to the decommissioning one. The FMEDA takes place after the concept phase and before the production of the item. It has the purpose to allow the designers to determine if the reliability of their item is compliant with the minimum reliability requirements determined during the concept phase and described in the *technical safety concept*, which describes how to obtain the required safety level from a technical perspective.
These requirements are imposed by the Standard based on the results of the Hazard Analysis and Risk Assessment (HARA) performed during the "Concept phase". All the risks related to the item operations have to be analyzed, by assessing the most common operational situations [2][3][4]. At the end of this assessment, an Automotive Safety Integrated Level (ASIL), that is the formal indication of the risk level associated to a violation of a safety goal, is determined. If the item has more than one safety goals, the item is developed with the prescription required to the higher ASIL. ASILs are represented by a capital letter, from A the less restrictive to D the most severe. If the violation of a safety goal has no consequences on safety, it is indicated as Quality Management (QM). In this case, it has not to be considered within the safety lifecycle but has to be managed

under the quality management procedures adopted by the manufacturer company.

Once we have obtained the ASIL for the item, the design will have to comply with the relative hardware architectural metrics [1]. These are: random hardware fault metric $rhfm$, single point fault metric $spfm$, and latent fault metric lfm. An accurate description of these is outside the scope of this work. The interested readers can find it in [10].

As required by the ISO26262, the expected results of the FMEDA process is a table in where, for each failure mode of each component installed on the board, is reported its failure mode effect. As described before, the possible failure mode effects can be classified into four classes, keeping into account if the detection system can identify them and if they affect safety.

- Safe Undetected (SU), when the considered failure mode has not relevant drawbacks in terms of safety and it is undetected. SU failures have to be avoided, since we are not investigating multiple failures conditions, hence we do not have any information about the effects of more than one SU failures at the same time.
- Safe Detected (SD), if the failure mode has not relevant drawbacks in terms of safety and it is detected by the failure detection algorithm.
- Dangerous Undetected (DU), when the failure mode has a dangerous effect and there are no detection mechanisms able to perceive it. Of course, these are the worst since they can be assumed as single point of failure.
- Dangerous Detected (DD), if the failure mode by itself could lead to dangerous effect, but it is possible to detect it. In this case, it is possible to insert a mitigation algorithm to mitigate its effects making it become an SD failure mode.

These classifications are usually performed by hand by safety engineers on the design to be analyzed. This method is effective to analyze those systems that have interactions with the physical environment (like anti-pinch systems for cars sliding windows or automatic parking brakes) for which the cause-effect relationship is clearly defined but, in those cases in where these interactions become more intricate, a perfect knowledge of the item behavior cannot be sufficient to determine the effect of a failure mode. Some aiding tools to improve the quality of the results have been proposed in the past [8][10][11], but they classify by applying rules keeping into account only the item(actuator)-level effects of the failure. For a lot of automotive functions, it can be sufficient, but in some cases, finding item-level classification criteria can become non-trivial, due to the strong coupling between the item local failure effects and vehicle behavior.

To overcome this issue, we would like to propose the following approach. Starting from the item simulation with fault injection methodology proposed in the previous works [10][11], we can determine the behavior of a fault-affected item. Hence, by the propagation of this behavior to a vehicle-level simulator, it becomes possible to assess also the expected effects on the whole car, allowing to define classification rules at the vehicle-level.

In this work, we adopted an actuator-based perspective: we are only considering the failures that propagate to the actuators. Hence, under this hypothesis, all the failure effects can be propagated from the inside of the item to the actuators without losing generality [5]. The core of a vehicle-level simulator is composed of a set of differential equations to simulate the vehicle dynamics and an optimized solver. A graphical user interface allows configuring scenarios, vehicle characteristics, and environmental conditions. They offer also visualization tools to analyze the simulation results as plots or 3D reconstructions. Usually is provided a quite good set of predefined simulations. High-end simulators offer also APIs to connect themselves with third-party software. In this way, it is possible to extract the signals of interest, compute the item behavior, and close the loop giving the actuators command to the simulator.

Other authors explored the simulation-based failure effects assessment, like [6] and [7] where a machine learning approach has been adopted to generate test cases able to shorten the identification of the dangerous faults.

III. Proposed Methodology

A platform suitable to perform FMEDA by the proposed vehicle-level simulation approach is composed of:

- the embedded software of the item;
- the physical models of the item (at printed board circuit level) in both fault-free and fault-affected conditions, needed to perform the SPICE-level simulations;
- the physical model of the sensors and the controlled actuators;
- the physical model of the car and the surrounding environment, provided by the vehicle-level simulator;
- scenarios in which test the failure effects;
- vehicle behavior classification rules.

The following accessory components are also required:

- fault list generator module;
- saboteur;
- circuit (item-level) simulator;
- vehicle-level simulator;
- failure effect classifier.

The whole system architecture is represented in fig. 1 (last page of the paper). For the sake of this work, we considered only the faults that could affect the analog components installed on the PCBs of the Powertrain Electronic Control Unit and of the two inverters. We have not considered the faults affecting both the wirings between the boards and the conductive tracks of the PCBs itself.

The environment works as follows. It starts with the bill of materials (BOM) and a fault catalog (for example [15], with probabilities computed as described in [14]). By combining these two documents it is possible to obtain the failure modes list to be used by the saboteur to inject the faults during the simulations. The circuit simulator takes the SPICE-level model of the item. It can be instrumented with dummy elements to allow to inject particular failure modes (for example switches to simulate open or short circuits). We have to provide also a workload list representing the conditions in where we want to assess the failure mode effects. The latter can be shared with the vehicle-level simulator. During the simulations, the circuit simulator interacts with the vehicle-level simulator, taking the input signals from the latter and generating the outputs for the actuators, closing the control loops the item is in charge of. At this point, the simulation results can be stored and classified, according to the classifications rules, taking into account the effects of the considered failure mode on the vehicle dynamics.

The saboteur injects one by one the faults [11][12][13] into the affected component (fault injection technique is widely discussed in the literature [16][17][18][19][20][21]). In the FMEDA only one failure mode is considered at a time, since the probability of successful detection is part of the analysis. The silicon-level faults that could affect the microcontrollers (MCUs) are not considered in this work since modern automotive-grade MCUs integrate fault detection and mitigation mechanisms [22][23] and have to be considered as Safety Element out of Context (SEooC) [1]. Hence, all the MCU related failures are described at PCB levels (like shorts between pins, soldering breaking down, and so on…).

IV. The Case Study

A. Simulation system set-up

To better describe the approach, we considered a powertrain system with a dual-motor axle. In particular, the safety of the software-implemented differential gear has been taken into account. The system is composed as follows: the rear axle of the vehicle is powered by two independent motors, one for each wheel. This allows saving weight, since no shafts and differential gears between the two sides of the car are needed, and each motor has to produce only half of the power. It allows also a better control on the vehicle since the torque on the wheels can be varied to take into account the radius of curvature that the driver intends to travel, in a differential-drive like fashion. As a counterpart of all these interesting characteristics, such a system needs to guarantee a high level of reliability, since a failure on one motor can cause torque disparity between the two sides of the vehicle, making it very difficult for the driver to keep control of the traveled trajectory. The embedded software is in charge to detect failures on the motors and it must be able to take action to minimize the torque disparity.

The benchmark application is composed of (see Fig. 1):

- the embedded software of the dual-inverter system;
- the physical model of the inverters, with their fault model to be injected;
- the physical model of the motors;
- the physical model of the car and the surrounding environment, provided by the vehicle-level simulator;
- scenarios in which test the failure effects;
- vehicle behavior classification rules.

The following companion components are also required:

- fault list generator module implemented as a MathWorks™ MATLAB™ script;
- saboteur implemented as a MATLAB™ script;
- circuit simulator, resorting to MathWorks™ Simulink™ with SimScape™ toolbox, to perform the SPICE-level simulation of the design;
- an off-the-shelf vehicle-level simulator;
- failure effect classifier implemented as a MATLAB™ script.

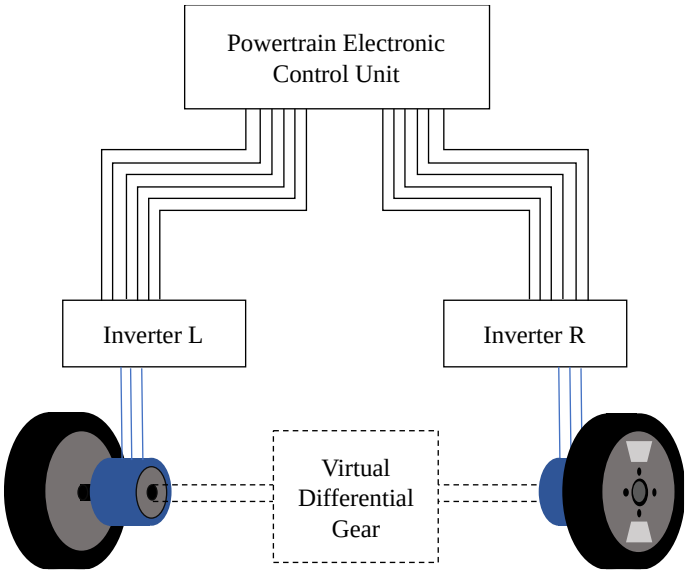

Fig. 1. Structure of the rear dual-motor axle of the car. The differential gear and the shafts between the two wheels are not physical devices but they are implemented by the embedded software.

1) Detection and mitigation algorithms

We identified 68 failure modes for the considered item. Of these, 30 regarding the gas pedal position acquisition chain circuitry, 2 the power supply, and the remaining 36 are about the two motors actuation chains. Hence, for each motor, we have 18 possible failure modes, 6 regarding the triple redundancy encoders installed to monitor the wheel speed and 12 the power electronics. An inverter is composed of 6 Insulated Gate Bipolar Transistors (IGBTs), and each one of them can remain stuck at closed (short-circuit) or open condition. Since in case of a short circuit of an IGBT, no software mitigation solution is possible (the fuses will melt down disconnecting the phase from the battery) we injected only an always open circuit failure on an IGBT of the left motor. This injection is sufficient to cover all

the possible cases due to the symmetries of the considered system.

As part of the benchmark application, we developed simple detection and mitigation algorithms.

The detection algorithm is based on a comparison between the current on one of the three phases, with the min/max values of the other two. If the disparity is higher than 80%, a fault into the considered leg of the inverter is detected. If the disparity is on the maximum values, the failed IGBT is the one connected to the positive pole of the battery, otherwise, it is the one connected to the negative pole.

The mitigation strategy is based on the following assumption: the motor driven by the inverter with the broken IGBT is not more able to provide the full torque it is expected to produce. Hence, since only one of the two motors is affected by this failure, and the most dangerous situation is caused by the asymmetrical torque, we could intervene on the fault-free motor. If we threshold its speed setpoint up to the speed of the fault-affected one, we can limit the torque disparity on the wheels. A semi-formal representation of the algorithm is shown in Fig. 2. *UpperLimit* and *LowerLimit* signals are equal to the speed of the fault-affected motor, *Detection* comes from the detection algorithm, and *NormalReference* is the speed request from the driver. Once a failure is detected, the *Detection* input signal is put to *true* and the driver's speed request is saturated up to the *UpperLimit* in case of forwarding direction or *LowerLimit* in case of reverse direction. The algorithm remains in the safe state, even if the detection algorithm stops to perceive a failure.

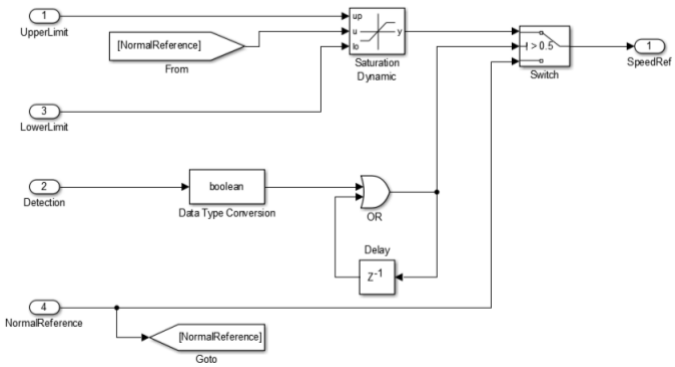

Fig. 2. Semi-formal (MathWorks Simulink™) model of the mitigation algorithm.

B. Simulation results

The key point of the proposed approach regards how to improve the evaluation of the effects of failures on the vehicle drivability. In the analyzed system, a disparity in the torque can cause a sudden turn of the car. The embedded software has to be able to "trim" automatically (in a similar way it is done on dual-engine aircrafts) the torque. Anyway, since the risk level associated to a vehicle function is determined to keep into account also the capability of an average driver (defined as *controllability* by the ISO 26262) to mitigate the failure effect, we represented the average driver as a PID controller (to keep into account the human reaction time), with a target behavior represented by a predetermined trajectory to be followed by the vehicle.

We simulated these conditions, representative of some significative operational conditions of the vehicle:

a) driving straight at 130 km/h;

b) acceleration from 0 to 130 km/h;

c) triple curving at 100 km/h;

d) regenerative braking on a straight road from 130 km/h to 0 km/h;

e) regenerative braking on triple curving from 100 km/h to 0 km/h.

Among these cases, the three most interesting are the b), c) and e). In the other two cases, a) and d), the results with and without the mitigation obtained are too close to each other that any discussion on the results is not possible. In any case, in order to report some information, we collected those simulation results in table 1. The reported values are the local maximums and minimums after the detection of the failure is happened.

TABLE I. EXPERIMENTAL RESULT SUMMARY

CASE	LATERAL ERROR [M]	YAW ANGLE ERROR RANGE [DEG]
A)	[-0.000 , 0.000]1	[-0.000 , 0.000]1
	[-0,117 , 0.100]2	[-0.720 , 0.385]2
	[-0.096 , 0.080]3	[-0.289 , 0.248]3
D)	[-0.000 , 0.000]1	[-0.000 , 0.000]1
	[-0.026 , 0.032]2	[-1.477 , 0.813]2
	[-0.026 , 0.031]3	[-1.453 , 0.796]3

1 Fault-free conditions.

2 Fault condition.

3 Fault condition with mitigation algorithm enabled.

1) Acceleration from 0 to 130 km/h

The first situation we simulated is the acceleration from 0 to 130 km/h. The car took 16 s in the fault-free condition to reach the target speed.

As shown in fig. 3, the mitigation algorithm is quite good to limits the failure effects in terms of lateral error. Since the failure is detected (at about 2 s from the start of the simulation) the benchmark mitigation algorithm can limit the lateral error, especially at high speed (when it is more difficult for a human driver to intervene).

Analyzing the error in terms of yaw angle (see fig. 4), we can see that the mitigation algorithm is able to reduce the error from the range from -0.5 to 0.6 deg to a range of -0.2 to 0.2 deg.

So, in this case, the mitigation algorithm, even if it is really simple, has demonstrated itself able to reduce both the lateral and the yaw angle errors of the car.

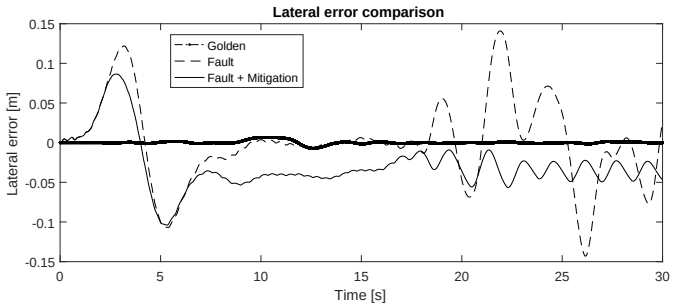

Fig. 3. Lateral error comparison with respect to the desired path (lane centerline) in fault-free, fault affected, and fault+mitigation conditions.

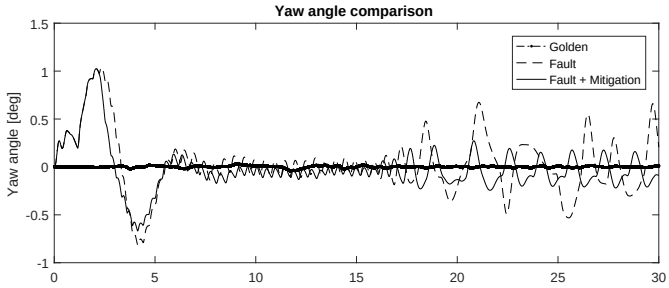

Fig. 4. Yaw angle error comparison with respect to the vehicle perfectly aligned with the road center line into fault-free, fault-affected, and fault+mitigation conditions.

2) Triple curving

In the straight acceleration, we obtained quite good results from the chosen algorithm. So, to keep into account a different condition, we repeated the experiments on a curving track, shown in fig. 5.

As shown in fig. 6 and fig. 7, the mitigation strategy adopted improves the lateral error and worsens the yaw angle performances. But this is an expected result since we are bounding the speed of the fault-free wheel to the fault-affected one. In any case, the error is low due to the chosen speed-control strategy that adopts a small proportional gain in the speed controller.

3) Regenerative braking on triple curving

In this case, we can see that the mitigation algorithm does not improve the lateral error (fig. 8) and worsen the yaw angle (fig. 9). In any case, these errors are inside the acceptable range, so in a tradeoff it remains convenient the mitigation algorithm adoption.

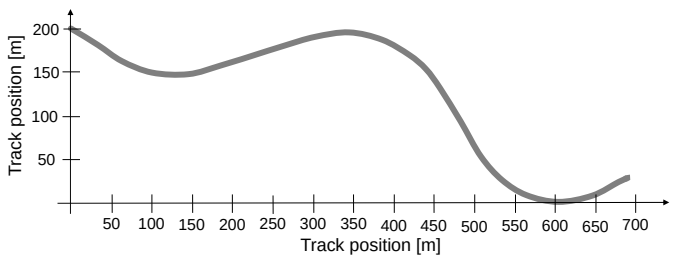

Fig. 5. The triple curving track implemented in the simulation enviroment.

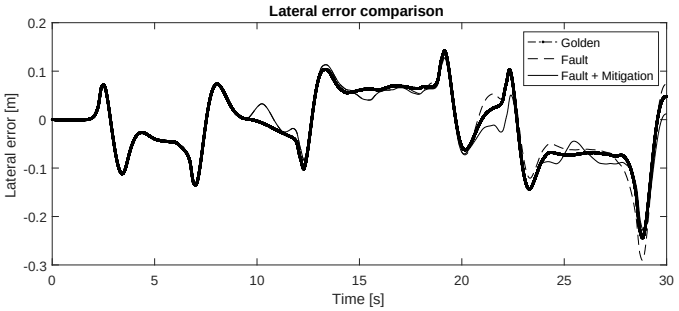

Fig. 6. Lateral error comparison with respect to the desired path (lane centerline) in fault-free, fault affected, and fault+mitigation conditions.

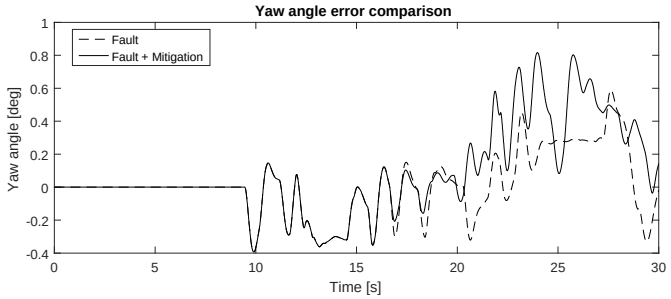

Fig. 7. Yaw angle error comparison, with respect to the path in fault-free condition, in fault affected and fault+mitigation conditions.

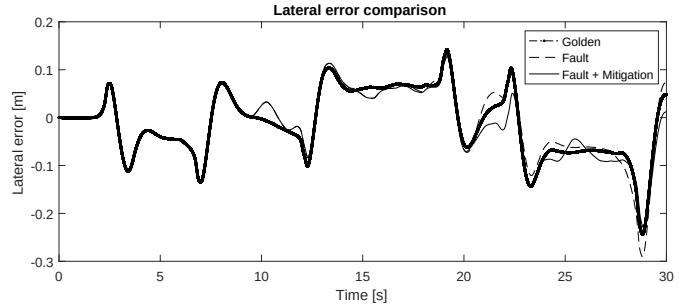

Fig. 8. Lateral error comparison with respect to the desired path (lane centerline) in golden, fault, and fault+mitigation conditions.

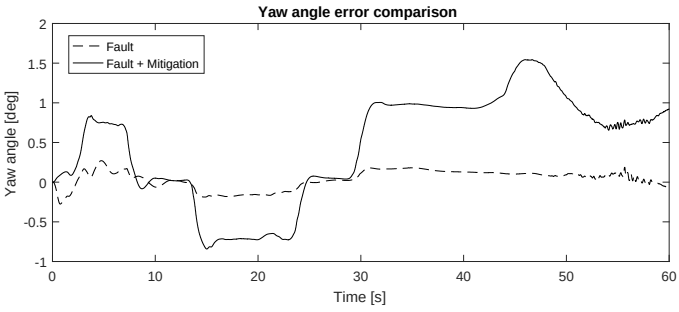

Fig. 9. Yaw angle error comparison, with respect to the path in fault-free condition, in fault and fault+mitigation conditions.

V. CONCLUSIONS

The purpose of this paper is not to propose a good set of detection and mitigations algorithm to be applied to a dual-motor axle, but to propose a framework to aid safety engineers who have to deal with FMEDA. Due so, we decided to show the simulation results and not an FMEDA table with the failure modes effects classification. Simulation-based approaches are widely adopted during the development of the automotive software, but usually only to verify in the early stages if the system can reach the nominal performance requirement. We proposed to introduce this approach also during the safety analysis, to produce results that can aid these difficult phases.

We obtained some useful results about how to study the software/vehicle interaction by considering the application of two really simple detection and mitigation algorithms on a dual-motor axle, so the approach demonstrated itself able to aid the functional safety engineers.

REFERENCES

[1] ISO 26262-10:2012, Road vehicles - Functional safety

[2] Koopman P., Wagner M., "Autonomous Vehicle Safety: An Interdisciplinary Challenge", In: IEEE Intelligent System Transportation Systems Magazine, 90-96, Spring 2017

[3] Jang. H.A., Kwon H.M., Hong S., Lee, M. K., "A study on Situation Analysis for ASIL Determination", In: Journal of Industrial and Intelligent Information Vol. 3 No. 2, June 2015

[4] K Beckers, D. Holling, Coté I.M., Hatebur D., "A structured hazard analysis and risk assessment method for automotive systems – A descriptive study" In: Reliabilty Engineering and System Safety (2017) pg. 185-195

[5] Johanennessen, "Actuator Based Hazard Analysis for Safety Critical Systems", In: Computer Safety, Reliability, and Security SAFECOMP 2004 Proceedings

[6] S. Jha et al., "ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection," 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Portland, OR, USA, 2019, pp. 112-124. doi: 10.1109/DSN.2019.00025

[7] Saurabh Jha, Timothy Tsai, Siva Hari, Mike Sullivan, Zbigniew Kalbarczyk, Steve Keckler, Ravishankar K. Iyer (2018) "Kayotee: A Fault Injection-based System to Assess the Safety and Reliability of Autonomous Vehicles to Faults and Errors" In: Third IEEE International Workshop on Automotive Reliability & Test Research Area, https://research.nvidia.com/publication/2018-11_Kayotee%3A-A-Fault, last visited

[8] W. M. Goble (2010) , Control Systems Safety Evaluation and Reliability, third edition, International Society of Automation, ISBN: 978-1-934394-80-9

[9] N. Snooke and C. Price, "Model-driven automated software FMEA," in Reliability and Maintainability Symposium (RAMS), 2011 Proceedings - Annual, 2011, pp. 1–6.

[10] Bagalini, E.; Sini, J., Sonza Reorda, M; Violante, M.; Klimesch H.; Sarson, P.; (2017) "An automatic approach to perform the verification of hardware designs according to the ISO 26262 functional safety standard", 18th IEEE Latin America Test Symposium, Bogota, Colombia

[11] Sini, J.; Violante, M.; (2018) "An Automatic Approach to Perform FMEDA Safety Assessment on Hardware Designs", 24th IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS), Platja D'Aro, Spain

[12] R. Leveugle, D. Cimonnet, A. Ammari, "System level dependability analysis with RT-level fault injection accuracy", "The 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Cannes, France, October 10-13, 2004", IEEE Computer Society Press, Los Alamitos, California, 2004, pp. 451-458

[13] Sini, J.; Violante, M.; Dodde, V., Gnaniah, R.; Pecorella, L., "A Novel Simulation-Based Approach for ISO26262 Hazard Analysis and Risk Assessment, 25th IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS), Rhodes Island, Greece

[14] IEC 61709:2011, Electric components -Reliability - Reference conditions for failure rates and stress models for conversion

[15] FIDES website, http://fides-reliability.org

[16] A. Benso, P. Prinetto, "Fault Injection Techniques and Tools for Embedded Systems Reliability Evaluation", 2003, Kluwer.

[17] J. Arlat, Y. Crouzet, and J. C. Laprie, "Fault injection for dependability validation of fault tolerant computing systems," in 19th International Symposium on Fault-Tolerant Computing, 1989, pp. 348–355

[18] M. Vieira, H. Madeira, I. Irrera, and M. Malek, "Fault injection for failure prediction methods validation", in Proc. of Workshop on Hot Topics in System Dependability at DSN 2009, Estoril, Lisbon, Portugal.

[19] L. Grunske, K. Winter, N.Yatapanage, S. Zafar, and P. A. Lindsay, "Experience with fault injection experiments for FMEA", Softw. Pract. Exp., vol. 41, no. 11, pp. 1233–1258, Oct. 2011

[20] Cukier, M., Powell, D., & Ariat, J. (1999). Coverage estimation methods for stratified fault injection. IEEE Transactions on Computers, 48(7), 707-723

[21] Christmansson, J., & Chillarege, R. (1996, June). Generation of an error set that emulates software faults based on field data. In Fault Tolerant Computing, 1996., Proceedings of Annual Symposium on (pp. 304-313). IEEE.

[22] Hsueh, M. C., Tsai, T. K., & Iyer, R. K. (1997). Fault injection techniques and tools. Computer, 30(4), 75-82.

[23] D. Cotroneo & R. Natella (2013). Fault injection for software certification. IEEE Security & Privacy, 11(4), 38-45.

[24] Mariani, R., & Boschi, G. (2007, July). A systematic approach for failure modes and effects analysis of system-on-chips. In On-Line Testing Symposium, 2007. IOLTS 07. 13th IEEE International (pp. 187-188). IEEE.

[25] Eychenne C., Zorian Y. (2017). "An Effective Functional Safety Infrastructure for System-on-Chips", IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)

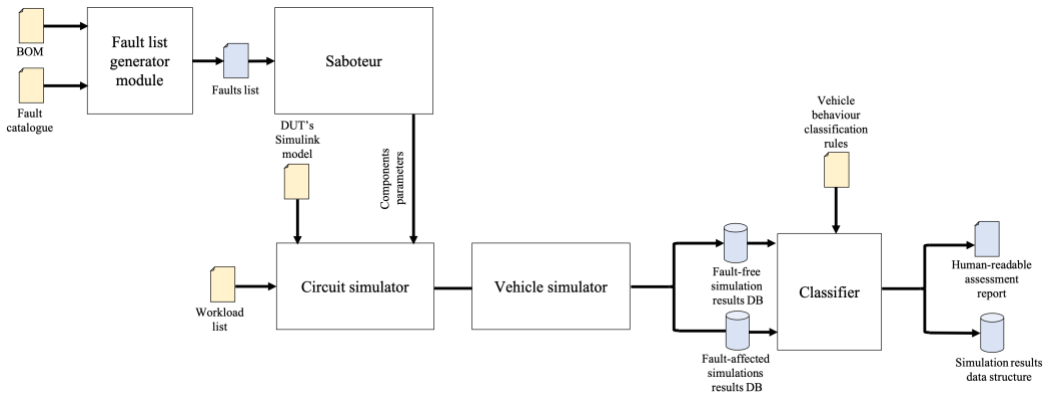

Fig. 10. Proposed methodology architecture