From Safety Analyses to Experimental Validation of Automotive Embedded Systems

Ludovic Pintard, Jean-Charles Fabre, Michel Leeman, Karama Kanoun, Matthieu Roy

From Safety Analyses to Experimental Validation of Automotive Embedded Systems

Ludovic Pintard [1,2,4], Jean-Charles Fabre [1,2], Michel Leeman [4],
Karama Kanoun [1,3], and Matthieu Roy [1,3]

[1] CNRS, LAAS, 7 avenue du colonel Roche, F-31400 Toulouse, France
[2] Univ de Toulouse, INPT, LAAS, F-31400 Toulouse, France
[3] Univ de Toulouse, LAAS, F-31400 Toulouse, France
firstName.lastName@laas.fr
[4] VALEO, 2 rue André Boulle, F-94000 Créteil, France
firstName.lastName@valeo.com

***Abstract*—Automotive embedded systems are becoming increasingly complex. Therefore verification activities are paramount to ensure safety. ISO 26262 is the first standard specifically dedicated to automotive safety systems. This standard requires introducing fault injection (FI) from the very early phases of the development process. Our work aims at developing an approach that will help integrate FI in the whole development process in a continuous way, from system requirements to the verification and validation phase. In this paper, we concentrate on exploring the benefits of safety analyses for experimental validation of the system. We propose an analogy between FI during the pre-implementation phase with safety analyses that are of common use during system design. We finally illustrate this approach on a case study from the automotive domain.**

Keywords— Safety; automotive embedded systems; ISO 26262 standard; safety analysis; FMECA; system validation.

I. INTRODUCTION

To deal with the criticality and the increasing complexity of electrical and electronic automotive systems, the automotive industry has introduced the ISO 26262 standard [1] for functional safety in road vehicle. This standard has pushed recommendations towards different methods and techniques that should be adopted in a development process to assure that "no unreasonable risk is due to hazards caused by malfunctioning behavior of electrical and electronic systems". In particular, ISO 26262 highly recommends fault injection (FI) as a technique to verify and validate electrical and electronic systems, throughout the whole development process, including the early design phase [2, 3]. It is worth to mention that, even though experimental validation of embedded systems is of common practice in industry, fault injection adoption in the early design phase, as advocated by the ISO26262 standard, is not common and unclear. The work presented in this paper addresses specifically our first investigations related to this challenging issue.

The purpose of this paper is to show that safety analyses are similar to fault injection in the early design phase of a system development. We show the strong link we identified between classical safety analyses, commonly used in industry for critical systems design (e.g. [4]), and fault injection principles at design phase. An even more challenging aspect is the smooth integration of FI in an existing development process without fully redefining it, in a complementary way with respect to safety analyses. We then show that fault injection in the pre-implementation phase leads to clearly identify more conventional FI experiments on the target system.

The novelty of this paper is that we focus on safety analysis to justify early stages fault injection in the ISO26262 standard. We underline the benefits of early analyses for experimental evaluation, but we do not go up to conventional fault injection experiments that are carried out at present.

This paper is organized as follows: Section II presents the system development process that is assumed along this paper. Section III discusses FI at pre-implementation phase while Section IV shows its role within the development process. The approach is then applied to a concrete case study from the automotive industry, in Section V. Section VI is devoted to related work, and Section VII concludes the paper.

II. SYSTEM DEVELOPMENT PROCESS

We assume a classical V-cycle development model, as depicted in Fig. 1. A system is decomposed into **products** that are in turn decomposed into **software blocks** and **hardware blocks** that are respectively decomposed into the elementary components of a system, **software modules** and **hardware parts**. The left hand side of the V is usually referred to as the design phase while the right hand side is usually referred to as verification and validation phase. In the context of this work, we use the terms **pre-implementation** and **post-implementation phases**, since the software modules and hardware parts are implemented only at the end of the pre-implementation phase. This figure does not show the relationships between the two branches of the V.

The goal of the development activities in the pre-implementation phase is the specification of the requirements a level should satisfy (levels are referred to as L0 to L3 in Fig. 1). These requirements are either functional or safety requirements. They are consistently refined from the highest level, L0 (system) to the lowest one, L3 (hardware parts or software modules). This figure separates the safety analyses

from the development activities for a clarification purpose. There are two main types of safety analyses: qualitative ones and quantitative ones, and their goal is to assess the safety of the proposed system. They allow verifying that the designed system fulfills the safety requirements. Modifications could be proposed to enhance safety at each architectural level to build a system that is "good and safe by design". The right hand side of the V cycle corresponds to the post-implementation phase, i.e. verification and validation activities of the target system.

The terminology used in the remainder of this paper is given in Fig 1; for instance, SW blocks defined at architectural level L2 are implemented as SW modules at level L3. The same applies to HW blocks defined at L2: they are composed of HW Parts at L3.

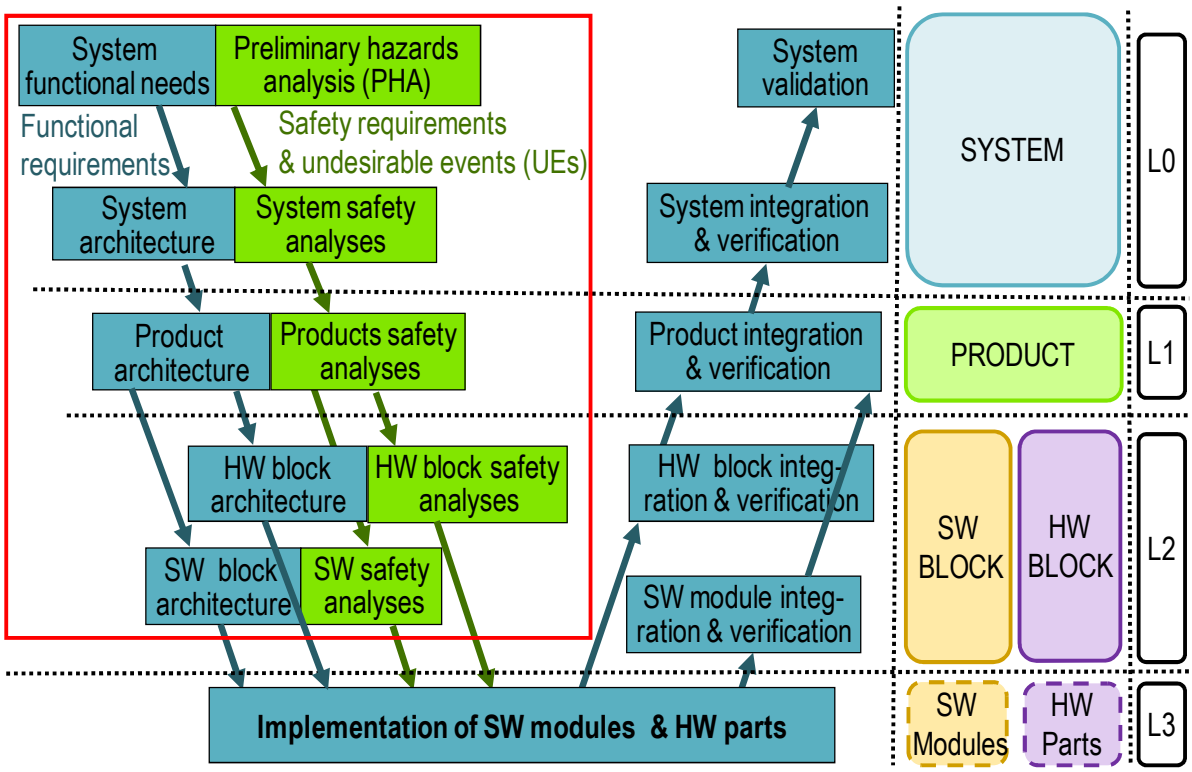

Fig. 1. V-cycle Development Process and Terminology Used

III. Fault Injection and Pre-Implementation Phase

FI is a technique commonly used for system validation. It is performed on a target system that is either a model of the system, a prototype or an instance of the developed system. The main elements that characterize FI (experiments, or campaign) are: the set of faults to be injected (the **F**ault model), the system activities under which the faults are injected (the **A**ctivation), the **R**eadouts of the experiment results, and the **M**easures evaluated, based on the readouts. These elements form the basis of the **FARM** [5] method to FI, similarly to other existing FI methods (see e.g., [6]). It is worth to mention that the measures, M, are defined first, since they guide the whole FI process, but their values are assessed in the last step, by processing information provided by the readouts. The fault model, the system activation and the readouts are defined in such a way that they enable obtaining the measures in a meaningful and efficient way.

The next section presents the applicability of the FARM method during the pre-implementation phase before concentrating on the FI flow for the product architecture level, L1, taken as an example. Then it explores the relationships between FI in the pre-implementation phase and other safety analyses, such as FMECA.

A. FARM Method in the Pre-Implementation Phase

During the pre-implementation phase, functional requirements at various levels are provided in the form of textual descriptions or drawings giving clear definitions of the operation of the components involved at a given level (static view), or/and block diagrams that show the various interactions and dependencies between these components (dynamic view), etc. Models of system functions can also be built. Such models can be very brief, or more detailed ones, that can be executable, using software simulation tools, depending on the level considered.

In parallel, safety analyses (including for example Fault Tree Analyses, Minimal Cut Sets, Reliability Blocks Diagrams, Markov Chains, Failure Modes, Effects, and Criticality Analysis) are performed at the various levels.

In this context, the "targets" of FI correspond to functional requirements of: the system, the product, the hardware and software blocks, the software modules, and the hardware parts. A target at a given level Li is composed of lower-level (Li+1) functional requirements. Let us explore the meaning of the four elements of the FARM method in this context.

Measures: Without going into details and addressing measures specific to particular systems or to a given architectural level, the measures should allow to ensure that safety requirements are handled correctly, and the causes of safety requirements violation are mitigated as much as possible.

These very generic, high level, measures are to be refined with the architectural levels and all along the development process. A measure can be either qualitative (e.g., the property is true of false) or quantitative (in terms of probabilities, for example). Notice that, in the pre-implementation phase, we cannot estimate the distribution of failure modes of the system/component, but we can identify defense mechanisms that must be evaluated when implemented. We can also identify missing mechanisms leading to the violation of safety properties.

Fault Model: One possibility is to start with the failure modes of the architectural levels together with their potential causes. These failure modes are defined with respect to the safety requirements and to the measures to be analyzed. We consider that the potential causes of all failure modes at a given level form the set of faults of the considered level. In the pre-implementation phase, the fault model cannot be more precise than the potential causes that are identified at a given abstraction level. However, such potential causes can be triggered by low-level faults (i.e. software and hardware faults). During the post-implementation phase, several low-level faults can be identified as provoking a given potential cause identified during system design. Experiments can thus be optimized accordingly, selecting the fault to be injected according to the capabilities of the fault injection tool used.

Activation: The activation model of a component is related to its functional specification. This model describes when (and where) the faults should be injected. The definition of the system activation at this stage requires a description of the different activation modes of the target, some use cases, the dynamics of actions, etc. The more detailed is the modeling of the behavior at the considered level, the more relevant sequences or use cases can be defined.
It is worth to emphasize the prime role of behavioral models during the pre-implementation phase. They allow a thorough understanding of system function and behavior. In particular, they allow i) an easy identification of possible causes of failures, and ii) the analysis of fault propagation, in a precise way.

Readouts: During pre-implementation, readouts are specifically related to the states of the components as resulting from the propagation of the injected fault(s). They represent the effects of the application of the fault model on the target, assuming an activation model. For a given level, the readouts correspond to the set of observations that can be made at the considered level after injection of a fault. Therefore, the failure modes of the considered level are part of the readouts, but again probability distribution cannot be obtained through experiments in the pre-implementation phase.
The above shows that performing FI according to the FARM method is meaningful during the pre-implementation phase, provided that the various elements of the method are clearly defined for this phase. The main difference between FI during the pre- and post-implementation phases lies in the control of the fault propagation and in the nature of faults that can be injected.

- In the pre-implementation phase, fault propagation must be performed based on assumptions that are applied to the component functional descriptions either directly or using models, executable or not. In the latter case, model building requires a significant effort, but a tool should help to handle the complexity in order to perform the analysis faster. On the other hand, for post-implementation, fault propagation is directly related to the system activity and does not require any specific control.

- In the pre-implementation phase, one has to take into account all faults (or at least as much as possible) that may impact safety requirements, in order to analyze their effects and propose architectural solutions to reduce these effects. During post-implementation phase, not all faults can be injected easily, and a detailed analysis may be necessary to group faults with the same effects, in order to facilitate FI experiments.

In the rest of this paper, FI activities related to a system design (description or a model) will be referred to as "**Fault Injection Analysis", FIA**. Conventional FI techniques targeting the real system or a prototype will be referred to as **"Fault Injection by Experimentation", FIE**. With respect to Fig. 1, FIA corresponds to FI during pre-implementation and FIE to FI during the post-implementation phase.
This paper focuses on FIA. The rest of this section investigates the activities to be performed to produce a well-structured FIA for a given level. These activities, that are similar for all levels, are illustrated for the product functional requirements (level L1), considered as an example. A global approach that links the FIAs throughout the levels is given in Section 4.

B. FIA Flow for the Product Level, L1

Fig. 2 summarizes the different steps of the analysis and highlights the interactions of the FIA with the upper and lower levels L0 and L2, as well as with the adjacent analyses of Fig. 1 (product architecture, and other product safety analyses). The various boxes and their relationships are presented in the rest of this section.

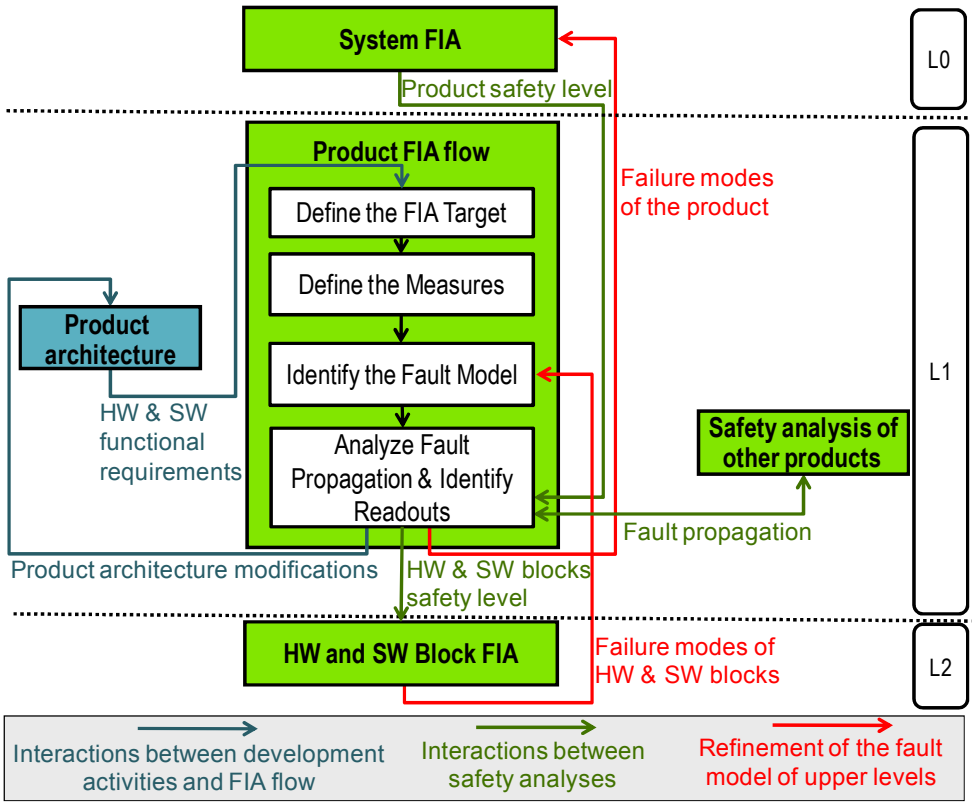

Fig. 2. FIA Flow of the Product Level and its Interactions with other Activities

The FIA Target corresponds to the HW and SW blocks functional requirements that are defined based on the product architecture requirements carried out at the same level, L1, (link from the Product architecture to the product FIA).

Measures Definition: A safety requirement is violated if an *Undesired Event, UE*, occurs at the system level. The violation of a safety requirement is usually measured in terms of probability of occurrence of the associated UEs. UEs are due to events occurring at the product or lower levels that propagate to the system level. The definition of these UEs allows identification of the possible failure modes of the product. In the automotive domain, each UE should be classified according to one of the four safety levels, referred to as **Automotive Safety Integrity Levels**, **ASIL**.

Safety mechanisms intended to i) detect possible failure modes (or their causes) and to ii) inhibit (or cover) their effects, are usually included in most of the architectural levels. Their aim is to handle the various failure modes in order to reduce the risk by diminishing the occurrence and effects of the UEs. Their efficiency corresponds to their error detection and recovery coverage (preventing a failure to occur and placing the system in a safe state), whose numerical values can only be evaluated based on experimentation during the post-implementation phase.

Fault Model Identification: This activity is related to the identification of the possible causes of the failure modes of the product, by determining the failure modes of HW & SW block functional requirements. The set of all potential causes of the failure modes of the product form the set of faults that have to be analyzed at the product level (or the fault model). The causes of the latter will be completed with information provided by the lower levels of architecture (via the link from the HW and SW block FIA, in Fig. 2). In other words, the analysis of both HW and SW blocks can help refining the faults that can impair the correct behavior of the product. This analysis can consider physical faults and software development faults that will be used later on to perform experiments.

Fault Propagation Analysis & Readouts Identification: For each failure mode, the knowledge of the product architecture and behavior should be used in order to propagate the failure modes of the HW&SW block to the product. The goal is to determine the effects of the failure modes of the system on the block itself and on the product, either performed by hand, or automated using an executable model (simulation of the propagation of failure). These effects are: i) local, i.e. the failure modes of the other HW&SW blocks of the product ii) the failure modes of the product that can propagate to other products (via the link fault propagation indicated in Fig. 2).

IV. FIA Through the Development Process

A. Relationships Between FIA and Other Safety Analyses

The various safety analyses, such as Fault Tree Analyses, Reliability Block Diagrams or cut sets, address the propagation of specific faults at each architectural level and between levels. Their ultimate aim is the identification of critical paths. This is also the aim of FI during the pre-implementation phase. More precisely, for a given level, FI and Failure Modes, Effects and (Criticality) Analysis FMECA [7] exhibit strong similarities and share several common objectives. An important objective shared between fault injection and FMECA is the identification of all critical faults/failures of the system. This objective is achieved by analyzing the effects of the potential causes of failures on the system in a FMECA, and by analyzing system behavior **in the presence of faults**. Both approaches are based on the same kinds of analyses. Moreover, both approaches aim at identifying components that require specific safety mechanisms (e.g., for error detection and/or error recovery mechanisms) to mitigate the effects of the critical causes.

Even though the FMECA is manageable manually at high levels (system or product levels), it becomes mandatory to use models and tools to support the FMECA at more detailed architectural and/or behavioral levels. This situation is similar for FI. Indeed, FMECA and FI analyses can be supported by the same kind of modeling formalisms and tools. In particular, behavioral models, based for example on Petri nets, finite state automata or even scenarios described with sequence diagrams, constitute a valuable support for both FMECA and FIA.

Finally, it is worth to mention that both FMECA and FI may be used for qualitative analyses purpose only (e.g., checking the non-existence of some specific failure modes), and both may also be used with quantitative analyses (to assess for example some parameters such as the proportion of potential causes leading to critical failure modes). Additionally, more precise values of such parameters may be obtained from FI (or other analyses) performed during the post-implementation phase. At early stages, the analysis can be assisted by tools when dynamic models are available at lower levels of the design process (e.g., sequence diagrams). The discovery of missing safety mechanisms can be an interesting outcome of FIA with this approach.

From a practical point of view, we can illustrate this analogy based on typical information reported in a FMECA spreadsheet, exemplified in Table I, in which an item is a function or a component.

TABLE I. Typical FMECA Spreadsheet Line

1	2	3	4	5	6	7	8
Item	Failure modes	Potential causes	Local effects	Upper-level effects	Risk level	Safety mechanisms (SM)	Upper-level effect with SM

A FMECA line is guided by the failure modes and their potential causes, assuming some activation modes of the system. However, the activation model is not shown in this line. It is usually provided by the underlying analyses performed to build the FMECA spreadsheets. A set of FI analyses, using as fault model the set of potential causes of failure modes, lead to the identification of the related failure modes of a component in a FMECA. A line of a FMECA spreadsheet can be seen as summarizing the results of a set of FI analyses. In other words, FIA makes visible and explicit the analyses supporting the FMECA spreadsheets.

It is worth to mention that what is usually called risk level in the FMECA spreadsheet corresponds to the **Safety level** (or ASIL level) introduced in Section 3.2.

B. Link Between FIA Levels

The various levels of the FIA can be linked based on the propagation of failure modes from one level to the upper level. Two links are of particular interest based on the following considerations:

- Link-A: level Li *failure modes* (Column 2) correspond to level Li-1 *Upper level effects* (Column 5).
- Link-B: level Li-1 *potential causes* (Column 3) correspond to level Li *failure modes* (Column 2).

These two links are at the origin of two types of causal chains: S-shaped and Z-shaped causal chains, depicted in Fig. 3.

Besides these two causal chains follow the traditional *fault-error-failure* model, as far as we are aware, these concepts have not been introduced in fault injection studies.

The S-shaped Causal Chain is based on Link-A. Li *failure modes* propagate to Li *upper level effects*, which correspond to an Li-1 *failure mode*, which in turn propagates to Li-1 *upper level effect*. An S-shaped chain:

- captures the propagation through the architectural levels of the effects of an initial failure mode, and provides traceability of the fault model with the safety level.
- enables the definition of safety mechanisms to handle the propagation at the most suitable architectural levels.
- helps in defining the readouts of FI experiments (FIE) at the successive architectural levels, during the post-implementation phase.

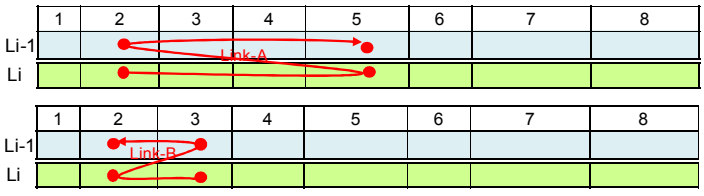

Fig. 3. S-shaped and Z-shaped Causal Chain

The Z-shaped Causal Chain starts from the *potential causes* column of level Li, and propagates as a *failure mode* of the same level, and continues on level Li-1 towards the *failure mode* column. A Z-shaped chain helps refining the fault model of the considered level by identifying equivalent faults (faults that lead to the same effects). Thus, it will help selecting the fault to be injected during FIE, taking into account the FI instrumentation and the FI accessibility of the target.

It is worth to mention that a failure mode may belong to more than one S- and Z-shaped chains, depending on the number of items in the lower levels contributing to this failure mode.

C. Concluding comments

We have shown that FI analyses during the system pre-implementation phase provide information that can be synthesized in FMECA spreadsheets. The advantage of FIA is to make visible and explicit all the detailed analyses performed manually or based on models and tools for activation and for fault propagation analyzes.

In order to conform to the ISO 26262 standard, many actors in the automotive domain have started looking for new approaches to FI in the early development phase, with the fear that integrating FI in the early development process will incur a redefinition of the whole development process. Interestingly, the analogy between FIA and FMECA developed in this paper shows that FI can indeed be easily integrated in the existing process, and will even improve the efficiency of the global process. In particular, according to ISO 26262:

- FMECA must be integrated in the development process and, in practice, FMECA is already integrated.
- Safety analyses are required at system and product level (ISO 26262, Part 4), HW level (ISO 26262, Part 5) and also SW level (ISO 26262, Part 6).

Thus, the integration of FIA will not incur additional costs with respect to the integration of FMECA. However, as emphasis is put from the beginning on failure propagation, based on the exploitation of the S- and Z-shaped chains, these early analyses will be very helpful for:

- managing failure propagation from the early development phases, in order to recommend safety mechanisms, at the right locations in the system.
- planning FI experiments in the post-implementation phase.

Finally, despite their prime importance in failure mode propagation analysis, as far as we are aware, the S- and Z-shaped chains have not been highlighted in early studies. They are one of the main benefits brought about by FIA.

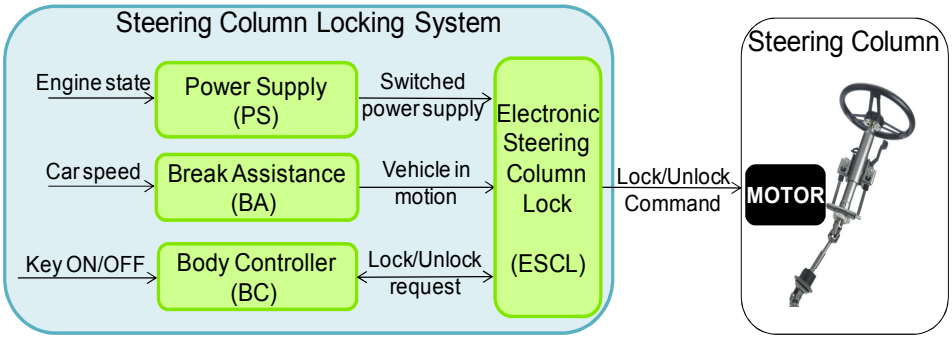

Fig. 4. Steering Column Locking System Architecture

V. Case Study – A Steering Column Locking System

We illustrate our approach on a Steering Column Locking System that controls a locking/unlocking motor on the steering column of the car, presented in Fig. 4.
This system has two functional requirements:

- The locking management of the steering column when the driver wants to immobilize the vehicle and prevent theft of the vehicle.
- The unlocking management of the steering column when the driver wants to move off.

There are two system safety requirements, referred to as Safety Goals (SG) that must be ensured. Each SG is allocated an automotive safety integrity level (ASIL):

- **SG1**: The system shall not lock the steering column when the vehicle speed is over a pre-defined threshold. It has the highest safety level, ASIL D.
- **SG2**: If the steering column is locked, the system shall prevent to start the engine of the vehicle. SG2 has the lowest safety level, ASIL A.

In the rest of this section, we will first illustrate the FIA approach at system level L0, to identify failure modes that could violate the safety goals SG1 and SG2, and to check the existence of safety mechanisms preventing their occurrence. Then, we will use results from the FIA on product level (L1) to illustrate the S-shaped chain. Current and future work is summarized at the end of this section.

A. Steering Column Locking System FIA (L0)

FIA Target: *Products Functional Requirements*. The products must ensure the system functional requirements and take into account system safety requirements. The product functional requirements are allocated to the architecture given in Fig. 4. They are extracted from the System Architecture Activity (Fig. 1) and summarized in Table II. The ESCL is the main product in the system. The three other products provide common functions such as energy, information for safety purpose ("vehicle in motion" information) or centralized information from the driver (the Body Controller collects and transmits commands from driver's interfaces).

TABLE II. FUNCTIONAL REQUIREMENTS OF THE PRODUCTS

Product	Functional requirements
ESCL: Electronic Steering Column Locking	**ESCL-F1**: Lock the Steering column **ESCL-F2**: Unlock the steering column
BC: Body Controller	**BC-F1**: Transmit Lock Command from driver's interfaces to ESCL **BC-F2**: Transmit Unlock Command from driver's interfaces to ESCL
BA: Break Assistance	**BA-F**: Transmit "vehicle in motion" to the ESCL
PS: Power Supply	**PS-F**: Supply a switched electrical power to ESCL

TABLE III. FAILURE MODES OF ESCL

Functional requirement	#	Failure mode	Product effects
ESCL-F1	FM1	Spurious Lock	Erroneous lock command
	FM2	ESCL-F1 Lost (No lock)	No lock command is possible
	FM3	ESCL-F1 stuck-at	ESCL always performs lock command
ESCL-F2	FM1	Spurious Unlock	Erroneous unlock command
	FM2	ESCL-F2 Lost (No unlock)	No unlock command is possible
	FM3	ESCL-F2 stuck-at	ESCL always performs unlock command

Measures: Two main measures are to the probability of violation of SG1 and SG2.

Failure Modes: Table III lists the failure modes of the ESCL and their local effects. This table is obtained by analyzing functional requirements of ESCL-F1 and ESCL-F2, as well as the propagation of the failures at product level.

Fault Propagation and Readouts Identification:
We focus on the fault propagation of the ESCL-F1-FM1: spurious transmission of a lock command when the vehicle is at *high speed* leads to *Steering column locked* as a local effect while the speed is over the pre-defined threshold. The result at system level is the *locking of the steering column by the ESCL while driving*. The system effect violates the safety goal SG1.

The FIA aims at checking the existence of (or defining) safety mechanisms to prevent this propagation and the violation of SG1. Two safety mechanisms are identified:

- Braking Assistance product sends vehicle in motion signal to the ESCL when the speed is higher than a threshold, thus an ESCL mechanism must check this value before locking the motor. If vehicle in motion is true then SSM1 must inhibit lock command. (SSM1)
- The Power Supply product is a safety mechanism, as it must not power the ESCL when the car engine is running, thus the ESCL is in a safe state. (SSM2)

The FIA of the ESCL-F1-FM1 results in the first line of FMECA in Table IV. Similarly, analyzing the others failure modes, we obtain the complete System FMECA table, including the excerpt shown in Table IV.
Thus FIA identifies three safety mechanisms (SSM1, SSM2, SSM3) whose coverage will be measured through experiments on real target components using conventional fault injection. The failure modes will be used to select the most appropriate faults to be injected, i.e., fault that should be handled by each safety mechanisms.

B. ESCL Product FIA Flow (L1)

To illustrate our approach, we concentrate on the ESCL product. The HW&SW Blocks functional requirements, allocated to the product architecture of Fig. 5, are:

- Micro-controller Block: it controls the state of the MDB, and communicates ESCL Status through Communication Block.
- Communication Block: it transmits requests from BC and replies from ESCL.
- Motor Drive Block (MDB): it powers the motor of the steering column. This power converter output is controlled by the micro-controller using four switches: locking, unlocking, braking the motor, and un-power the motor.
- Sensor Block: it senses the position of the motor of the steering column (locked, unlocked, unknown).

Measures: We have identified three critical failure modes (or UEs) at system level: Spurious lock (ESCL-F1-FM1), ESCL-F1 stuck-at (ESCL-F1-FM3), No Unlock (ESCL-F2-FM2). At this level, the causes of these UEs should be identified.

In this section, we focus on one critical failure mode: "Micro-Controller-F1-FM1, *Erroneous assignment of the outputs of the micro-controller* delivered to the MDB" given in Table V.

The Micro-Controller-F1-FM1 failure mode puts the MDB in a locking state, the MDB powering up the motor in locking mode. In this case, the ESCL triggers a spurious lock of the steering column (ESCL-F1-FM1 failure mode). A detection mechanism (based on a sensor) should be added in the MDB to sense its current state. The detection mechanism should trigger a recovery action within the micro-controller, stopping the MDB when such inconsistent state is detected. These error recovery mechanisms deactivate the MDB and thus bring the ESCL in a safe state.

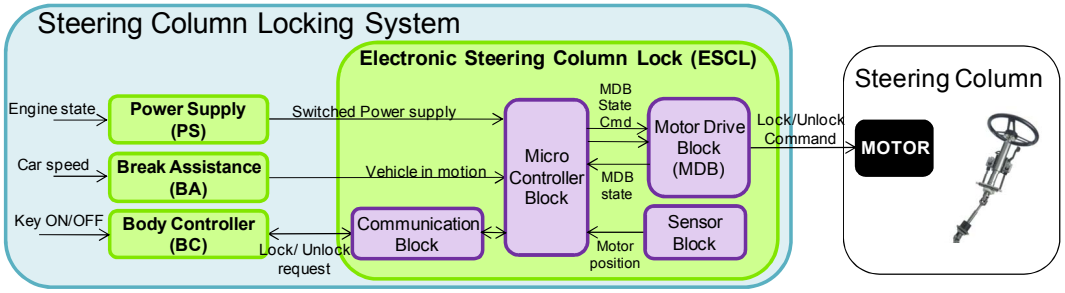

Fig. 5. HW and SW Blocks at ESCL Product Level

TABLE IV. PARTIAL FMECA OF THE STEERING COLUMN LOCKING SYSTEM: FOCUS ON ESCL PRODUCT

Item	Failure Modes	Potential Causes	Local effects	Upper-level effect	Safety level	System Safety Mechanisms (SSM)	Upper-level effect with SSM
Lock steering column ESCL-F1	Spurious Lock ESCL-F1-FM1		Erroneous lock command	Steering column locked while driving SG1 Violated	ASIL D	**SSM1:** Vehicle in motion **SSM2:** Switched power supply	No effect *
	ESCL-F1 Lost (No lock) ESCL-F1-FM2		No lock command is possible	Parked vehicle with steering column unlocked		NA	
	ESCL-F1 stuck-at ESCL-F1-FM3		ESCL always performs lock command	Steering column remains locked => vehicle starts with locked column SG2 Violated	ASIL A	**SSM3:** Monitoring of motor position should be implemented	No effect *
Unlock steering column ESCL-F2	Spurious Unlock ESCL-F2-FM1		Erroneous unlock command	Parked vehicle with steering column unlocked		NA	
	ESCL-F2 Lost No unlock) ESCL-F2-FM2t		No unlock command is possible	Steering column remains locked ==> vehicle starts with locked column SG2 Violated	ASIL A	**SSM3:** Monitoring of motor position should be implemented	No effect *
	ESCL-F2 stuck-at ESCL-F2-FM3		ESCL always performs unlock command	Parked vehicle with steering column unlocked		NA	

* assuming a perfect coverage of safety mechanisms

TABLE V. PARTIAL FMECA OF THE ESCL (ONE FAILURE MODE OF THE MICRO-CONTROLLER BLOCK).

Item	Failure Modes	Potential Causes	Local Effects	Upper-level effect	Safety level	Safety Mechanisms (SM)	Product effect with SM
Control the state of the MDB Micro-Controller-F1	Erroneous assignment of outputs of the micro-controller Micro-Controller-F1-FM1		Spurious activation of the MDB locking state.	Spurious lock ESCL-F1-FM1	ASIL A (D)	**SM1:** MDB Sensor => The detection of an error must lead to shutdown the ESCL (safe state)	No effect *

* assuming a perfect coverage of safety mechanisms

This example illustrates the links between the product and system levels of the architecture, thanks to the S-shaped chain (see section 4.2) whose elements are indicated in red in Tables IV and V. By way of example, during the pre-implementation phase, this chain helps in the following two activities:

- **Propagation through the architectural levels of the effects of an initial failure mode and traceability of the fault model with the safety level.**

  The failure mode Micro-Controller-F1-FM1 leads to the ESCL-F1-FM1 product failure mode that impacts SG1 at system level.

- **Definition of safety mechanisms to handle error propagation at the most appropriate architectural levels.**

  The proposed design includes safety mechanisms (SM) at two levels. SM1 detects and recovers at product level Micro-Controller-F1-FM1 failure mode. However, if SM1 fails, ESCL-F1-FM1 occurs but can be covered by SSM1 and SSM2 at system level. Indeed, the lack of coverage of SM1 could be handled by System level safety mechanisms. The two safety mechanisms placed at two levels are motivated by the high safety level required.

C. *From FIA to Experimentation*

Our aim in this section is to answer the following question: *To what extent FI analyses in Section 5.2 are of interest for conducting FI experiments in real targets?*

The various attributes of the FARM model, introduced in Section 3, have to be defined as follows:

Faults: a set of faults that aim at triggering the safety mechanisms must be defined. Through the S-chains obtained in the pre-implementation phase, the practitioner can easily identify the set of faults to be injected for each safety mechanism. Fault injection is sometimes misleading for non-specialists, since errors are injected and not faults. We do not want to dive here into terminology aspects, but in reality errors are injected and not faults, the fault being the supposed cause of the error affecting the system state. An error like the Micro-Controller-F1-FM1 can be the effect of many HW and SW faults, according to the causal chain fault-error-failure. The job of the practitioner is to find the easiest way to provoke the error, e.g. the "*erroneous assignment of outputs of the micro-controller*" in Table V. The error can be injected using physical probes or SWIFI depending on the accessibility of the value to be corrupted. In the case study example given in Table V, the corruption of the output value can be implemented in both ways: an access to the physical link between the microcontroller and the MDB, a corruption of the outputs using JTAG probes or even by mutation of the software instructions delivering the output to the MDB: the simplest the better in terms of implementation complexity but also in terms of reproducibility. FIA helps thus in defining faults to be injected in experiments targeting real components.

Activity: the activity during the fault injection campaign in the post-implementation phase should activate the target system according to some application profile. The application profile usually stands in well-identified scenarios covering ideally the whole state graph of the target component/system. Following our example, the valid activation profiles correspond to the set of possible execution flows leading to the delivery of an output of the micro-controller. This means that the practitioner should take into account the combination of input values triggering the activity of the micro-controller (i.e., *switched power supply, vehicle in motion, requests from the Body Controller, motor position*) and delivering an output to the MDB. Although, the runtime behavior is not detailed at early stages of the design, going closer to the implementation during the design phase helps understanding the behavior of implemented components through dynamic models (Petri Nets, state automata or simple sequence diagrams). When the system simple (at least in our case study), a set of few sequence diagram analyzed in the FIA helps defining the various activation profile of the micro-controller. Indeed, the number of valid combinations of input values is small. Again, this shows that this is something that can be extracted from the FIA to help the practitioner in setting up the FI experiments on real targets.

Readouts: obtaining the right set of readouts is essential to validate the safety mechanisms and to compute the measures. A bad identification of the readouts can have to side effects: i) a safety mechanism is not triggered whereas the system state is corrupted; ii) the safety mechanism is activated whereas the system state is correct. In our example, from the FIA Tables IV and V, one can easily identify the readouts to be captured to validate a safety mechanism. Reading the tables the practitioner knows that the following items must be captured: lock command, vehicle in motion, switched power supply for Safety Goal 1 (SG1).

Assessment of the Measures: In conventional fault injection experiments, the output of FI experiments is a pie chart composed of four main classes of outcomes: detected (error status, exceptions), not detected, recovered, not recovered. Ideally, 100% of errors are detected and recovered. In reality, some errors are not detected by internal error detection mechanisms or not recovered. Upper level safety mechanisms must then prevent the violation of safety properties in all cases. FI analyses check/recommend safety mechanisms to be put in place, FI experiments quantify their efficiency, namely detection and recovery coverage. When a safety property is violated the conclusion is two-fold: i) lack of coverage of a safety mechanism, ii) absence of a safety mechanism leading thus to a revision of the design.
We have shown in this section that practitioners conducting FI experiments on real targets can benefit from early stage combined analysis results. However, what we have proposed does not solve all the problems and some more work has to be done to set up fault injection experiments in practice. In particular, implementation details are needed to clearly identify the targets (SW or HW implemented modules), the faults that will really correspond to the potential causes leading to failure modes of a given target component, the probes required for both the injection of faults and the collection of readouts before computing the measures. This work is under progress on several case studies, in particular a core service in AUTOSAR [8], based systems, a generic complex Watchdog manager module [9] responsible for error detection (liveness, deadline, control flow errors) and for the triggering of recovery actions (like signaling and logging, partition reset, micro-controller reset and initialization) for automotive critical applications.

VI. Related Work

FI techniques have been traditionally performed on implemented targets to assess their fault tolerance or eliminate faults [10]. This research domain has been active for more than 20 years and many tools have been developed – see e.g. [11], [12], [13], [14].
FI has recently received renewed attention in the automotive domain since the settlement of ISO 26262, and for fault-injection benchmarking of embedded systems [2], [15].

In this paper, we tackle the problem of the relationship between FI during the early development phase, before system implementation, and FI on an existing system (i.e., an implemented target system). We have shown similarities between FIA and FMECA. The relationship between FMECA and FI on an implemented target system has already been partially highlighted in different works. For example, [16] states that Failure Modes and Effects Analysis (FMEA) should be complemented by fault injection experiments on hardware

parts to improve verification. [17] proposed a method to perform the verification at System-on-Chip (SoC) level, according to IEC 61508, using FMEA.

An aspect of our work is also to guide the definition of the fault model. The representativeness of this fault model is of a paramount importance to achieve effective FIE. Indeed, it must be defined carefully in order to obtain accurate and realistic results with a reasonable effort. [18] proposes an evaluation of the representativeness of software fault model, based on an extensive experimental campaign.

Besides, hierarchical FI has been investigated in DEPEND, a simulation-based environment, by injecting faults at several levels of abstraction [19].

VII. Conclusion

We have investigated how to integrate FI in the early phases of the development process of automotive critical systems, in the context of the ISO 26262 Standard. We explored the meaning and significance of FI analysis (FIA) during the pre-implementation phase, and showed the similarities between FIA and FMECA. In particular, we showed how FMECA spreadsheets can be used to synthesize the results of FIA. To consider FIA of successive development levels, we introduced S- and Z-shaped chains, linking the levels via their failure modes, their causes and their effects, to capture the failure propagation paths between the levels. The interest of these chains is twofold: i) they identify the nature and location of the safety mechanisms to integrate in the system, ii) they guide FI experiments during post-implementation phase.

We have shown the benefits of the proposed approach, which is compliant with the ISO 26262 standard, on a case study from the automotive domain. In this study, we outlined how top-level analysis in pre-implementation phase can help defining safety mechanisms and then defining the experiments to be carried out on implemented targets.

The ultimate aim of our work is to guide FI experiments (FIE), based on FIA. The main objective is to optimize the whole development process by defining an optimal set of experiments.

Our current work is dedicated to FIE implementation based on FIA. From a practical point of view, the FI tool developed at VALEO GEEDS provides all the functionality required to implement fault injection at various levels, electrical, physical, and using Lauterbach Nexus probe. The same features enable the readouts to be easily captured. This tool will be used in the next future to implement FI experiments on sizeable systems developed at VALEO following the approach proposed in this paper that is compliant with the ISO26262 standard recommendations.

To conclude, our main contributions in this paper are twofold: i) analysis of the relationship between two domains that are separated in practice, namely the failure mode, effects and criticality analysis and fault injection, and ii) the definition of the S- and Z-shaped chains to identify error propagation and to help making fault injection campaigns effective.

References

[1] ISO 26262 – Road Vehicles - Functional Safety, november, 2011. http://www.iso.org/iso/home/news index/news archive/news.htm?refid= Ref1499 [Online; accessed 30-Jul-2014].

[2] F. Ayatolahi, B. Sangchoolie, R. Johansson, and J. Karlsson, "A study of the impact of single bit-flip and double bit-flip errors on program execution," in Computer Safety, Reliability, and Security (F. Bitsch, J. Guiochet, and M. Kaaniche, eds.), vol. 8153 of Lecture Notes in Computer Science, pp. 265–276, Springer Berlin Heidelberg, 2013.

[3] R. Rana, M. Staron, C. Berger, J. Hansson, M. Nilsson, and F. Törner, "Improving fault injection in automotive model based development using fault bypass modeling," in 2nd Workshop on Software-Based Methods for Robust Embedded Systems (SOBRES'13), Koblenz, Germany, Sep 2013, 2013.

[4] Ann T. Tai, Chris J. Walter, Lorraine M. Fesq, John C. Day: Fault-class-aware fault tree generation and analysis. ISSRE (Supplemental Proceedings) 2013: 84

[5] J. Arlat, M. Aguera, L. Amat, Y. Crouzet, J. Fabre, J. Laprie, E. Martins, and D. Powell, "Fault injection for dependability validation: A methodology and some applications," Software Engineering, IEEE Transactions on, vol. 16, no. 2, pp. 166–182, 1990.

[6] J. Christmansson and R. Chillarege, "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, 1996.

[7] Department of the Army, TM 5-698-4, Failure Modes, Effects and Criticallity Analysis (FMECA) For Command, Control, Communications, Computer, Intelligence, Surveillance, and Reconnaissance (C4ISR) Facilities, 29 September 2006.

[8] AUTOSAR Development Cooperation, http://www.autosar.org. [Online; accessed 30-Jul-2014].

[9] AUTOSAR Specification of Watchdog Manager V2.2.0 R4.0 Rev 3, http://www.autosar.org/specifications/release-40/software-architecture/ system-services [Online; accessed 30-Jul-2014].

[10] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," Dependable and Secure Computing, IEEE Transactions on, vol. 1, no. 1, pp. 11–33, 2004.

[11] P. Yuste, D. de Andres, L. Lemus, J. Serrano, and P. Gil, "Inerte: integrated nexus-based real-time fault injection tool for embedded systems," in Dependable Systems and Networks, 2003. Proceedings. 2003 International Conference on, pp. 669–669, 2003

[12] M. Hsueh, T. Tsai, and R. Iyer, "Fault injection techniques and tools," Computer, vol. 30, no. 4, pp. 75–82, 1997.

[13] D. Skarin, R. Barbosa, J. Karlsson, "GOOFI-2: A tool for experimental dependability assessment," Dependable Systems and Networks (DSN), 2010 pp. 557–562.

[14] Giuffrida, C., Kuijsten, A., Tanenbaum, A.S., 2013. EDFI: A Dependable Fault Injection Tool for Dependability Benchmarking Experiments, Dependable Computing (PRDC), 2013 IEEE 19th Pacific Rim International Symposium on, pp. 31-40.

[15] N. Silva, R. Barbosa, J.C. Cunha, M. Vieira, "A view on the past and future of fault injection," Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on , vol., no., pp.1,2, 24-27 June 2013 doi: 10.1109/DSN.2013.6575332

[16] N. Bidokhti, "FMEA is not enough," in Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual, pp. 333 –337, jan. 2009.

[17] R. Mariani, G. Boschi, and F. Colucci, "Using an innovative soc-level FMEA methodology to design in compliance with iec61508," in Proceedings of the conference on Design, automation and test in Europe, DATE '07, (San Jose, CA, USA), pp. 492–497, EDA Consortium, 2007.

[18] Natella, R., Cotroneo, D., Duraes, J.A., Madeira, H.S., 2013. On Fault Representativeness of Software Fault Injection. Software Engineering, IEEE Transactions on 39, 80-96.

[19] M. Kaaniche, L. Romano, Z. Kalbarczyk, R. Iyer, R. Karcich, "A hierarchical approach for dependability analysis of a commercial cache-based RAID storage architecture", Fault-Tolerant Computing, 1998. Digest of Papers. pp.6-15