A Systematical Approach for "System Item Integration and Testing" in Context of ISO 26262

Martin Ringdorfer, Gerhard Griessnig(✉), Patrick Draxler, and Adam Schnellbach

AVL List GmbH, Hans List Platz 1, 8020 Graz, Austria
gerhard.griessnig@avl.com

Abstract. The development of automotive systems and components are often in context of a development of a safety critical system with respect to electric/electronic faults. To ensure that these systems are developed adequately safe, standards like the ISO 26262:2018 have evolved which should guide the engineers through the whole development. In this paper a possible testing approach for automotive safety critical systems on system level and on vehicle level is shown. Based on real-world examples, the approach for functional testing, fault-injection testing and robustness testing is investigated in detail. Several important steps which are needed to end up with a fully tested and calibrated safety critical product in context of ISO 26262:2018 are explored. Potential challenges that need to be solved during the testing/calibration activities are highlighted and discussed in detail.

Keywords: ISO 26262:2018 · Functional testing · Fault injection testing · Robustness testing · System and Item Integration and Testing · Safety validation

1 Introduction

The functionality of automotive control systems has increased in the past years. This has led to a continuous increase of complexity within electric/electronic (E/E) control systems. Since E/E systems rely on information sourcing, information processing and actuation behavior (e.g. sensors, control algorithms, electric actuation, …), single failures within the whole chain can cause inappropriate system reactions. In worst case, such failures might lead to unreasonable risks for the human life. From the engineering point of view, special attention must be put on such systems/functions in order to reduce the risk to an acceptable level. To identify such risks and to gain control over possible threads, more and more effort must be put into the development (implementation) and into the testing activities of such safety functions. This is where safety standards e.g. the IEC 61508 [1] or the ISO 26262:2018 [2] enter the game. For the automotive industry the functional safety standard ISO 26262:2018 is of interest, as it deals with the development of series production road vehicles. It is the task of such standards to define guidelines and recommendations on how to identify potential risks and on how to develop a safe product based on the best practices and state of the art.

Among all parts of the ISO 26262:2018, the part 4 is of special interest for this paper as this part defines how to test and validate safety critical control systems on system level and on vehicle level.

Referring to the SPI Manifesto [4], this paper addresses the principle 3 "Base improvement on experience and measurements" and the principle 6 "Use dynamic and adaptable models as needed" by updating of test model based on measured efforts for testing and gained experiences in executed industrial projects.

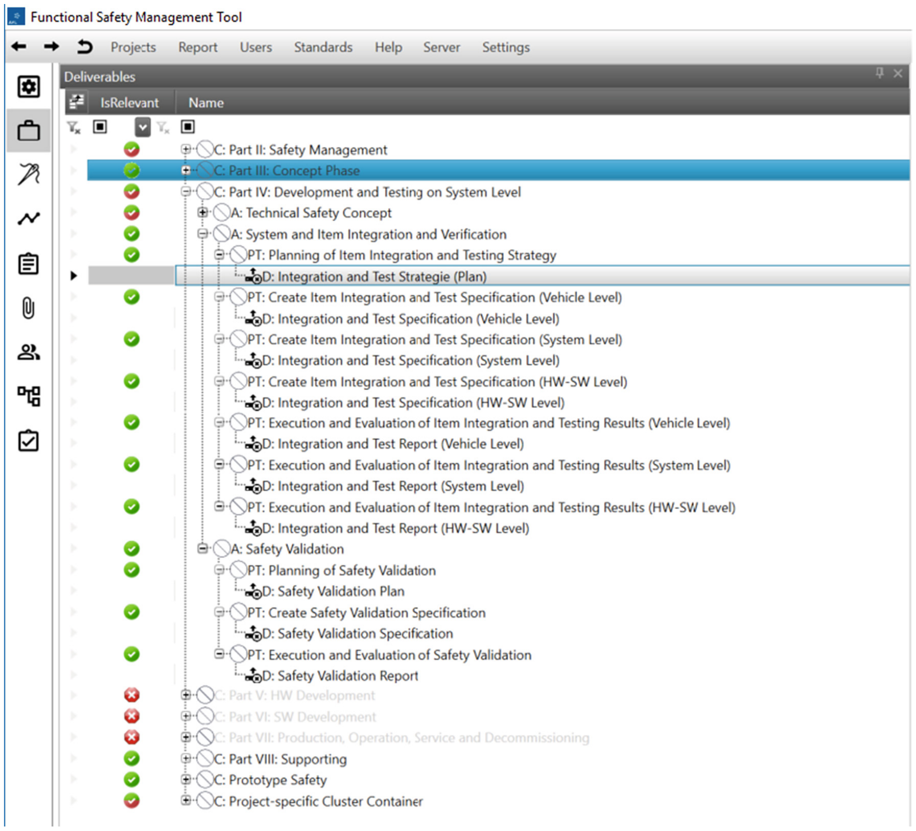
To give the interested reader an insight on testing of safety related systems, the paper is structured as follows: The first part gives an insight to the approach and to requirements as defined in the ISO 26262:2018 [2] in part 4 towards "System and Item Integration and Testing" (SIIT) as well as "Safety Validation" (SV). The second part explains two different approaches for function testing. Fault-injection testing (Does the device under test (DUT) show a proper safety reaction in case of E/E failures?) and robustness testing (Does the DUT not trigger a safety reaction when not needed?) are explained in detail. The experienced reader may have already be noticed that both methods might be contradicting with respect to calibration. This paper shows a potential way on how to establish a safe and robust system. Finally, a recommendation on the workflow for fault-injection and robustness testing is given and further improvements for future applications conclude the paper.

2 Requirements for Verification

The automotive functional safety standard defines many requirements towards organization, skills, processes, methods and expected evidences for the different phases of the product life cycle. The focus of this paper is on the verification and validation on system and vehicle level which are described in part 4 "Product development at the system level" of ISO 26262:2018. As the standard is written in natural language it is difficult for the reader to understand the systematic of the standard and the connections between the different safety activities. Therefore, a reasonable representation of the content is useful. For this purpose, the Fig. 1 from [3] shows the necessary safety activities which are requested for the "System and Item Integration and Testing" and "Safety Validation".

The objectives for the SIIT are defined by "(1) define the integration steps and to integrate the system elements until the system is fully integrated (2) to verify that the defined safety measures, resulting from safety analyses at the system architectural level, are properly implemented; and (3) to provide evidence that the integrated system elements fulfil their safety requirements according to the system architectural design" [2].

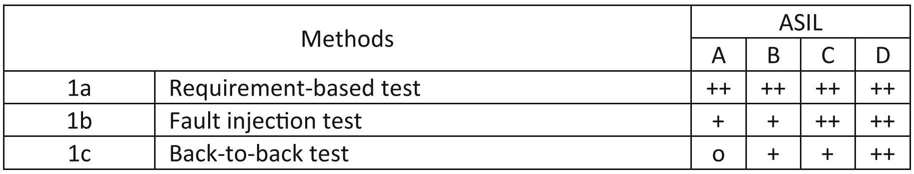
To achieve these objects the SIIT starts with the "Item Integration and Test Strategy". Within this plan the necessary test environments and equipment, the methods to derive test cases and the methods to perform tests shall be defined depending on the requested ASIL (Automotive Safety Integrity Level). This plan shall include the necessary integration steps and activities for HW/SW, System and Vehicle Level.



IsRelevant	Name
✓	C: Part II: Safety Management
✓	C: Part III: Concept Phase
✓	C: Part IV: Development and Testing on System Level
✓	A: Technical Safety Concept
✓	A: System and Item Integration and Verification
✓	PT: Planning of Item Integration and Testing Strategy
	D: Integration and Test Strategie (Plan)
✓	PT: Create Item Integration and Test Specification (Vehicle Level)
	D: Integration and Test Specification (Vehicle Level)
✓	PT: Create Item Integration and Test Specification (System Level)
	D: Integration and Test Specification (System Level)
✓	PT: Create Item Integration and Test Specification (HW-SW Level)
	D: Integration and Test Specification (HW-SW Level)
✓	PT: Execution and Evaluation of Item Integration and Testing Results (Vehicle Level)
	D: Integration and Test Report (Vehicle Level)
✓	PT: Execution and Evaluation of Item Integration and Testing Results (System Level)
	D: Integration and Test Report (System Level)
✓	PT: Execution and Evaluation of Item Integration and Testing Results (HW-SW Level)
	D: Integration and Test Report (HW-SW Level)
✓	A: Safety Validation
✓	PT: Planning of Safety Validation
	D: Safety Validation Plan
✓	PT: Create Safety Validation Specification
	D: Safety Validation Specification
✓	PT: Execution and Evaluation of Safety Validation
	D: Safety Validation Report
✗	C: Part V: HW Development
✗	C: Part VI: SW Development
✗	C: Part VII: Production, Operation, Service and Decommissioning
✓	C: Part VIII: Supporting
✓	C: Prototype Safety
✓	C: Project-specific Cluster Container

Fig. 1. Functional safety management tool [3]

After the planning activity the test case specification and finally the test case execution and evaluation will be performed (see Fig. 1). For the test case specifications and for the test case executions corresponding methods listed in tables (see Fig. 2) are defined within this standard.



Methods		ASIL			
		A	B	C	D
1a	Requirement-based test	++	++	++	++
1b	Fault injection test	+	+	++	++
1c	Back-to-back test	o	+	+	++

Fig. 2. Example table from [2] for methods for test case specification

The activities to achieve objectives for the SV which are “(1) to provide evidence that the safety goals are achieved by the item when being integrated into the respective vehicle(s); and (2) to provide evidence that the functional safety concept and the technical safety concept are appropriate for achieving functional safety for the item" [2] are similar to the SIIT (see Fig. 2).

It is not the intention of this paper to show that the derived testcases are complete or correct for the SITT and the SV, the focus is on the introduced systematic methodology to verify and validate that the product is safe and robust.

3 System and Item Integration and Testing

Common vehicle control function architectures (control system architectures) are structured in a multi-layer manner like shown in Fig. 3. In fault-free cases it is intended that sensor values or other input information are used by "nominal functions"[1] to determine actuator setpoints. Level 1 functions and calibration define the vehicle characteristic. Its focus is therefore e.g. the creation of a good driving behavior or the achievement of a low fuel consumption. In order to reach the target values, a high effort is spent to find a proper setting of the influencing Level 1 calibration.

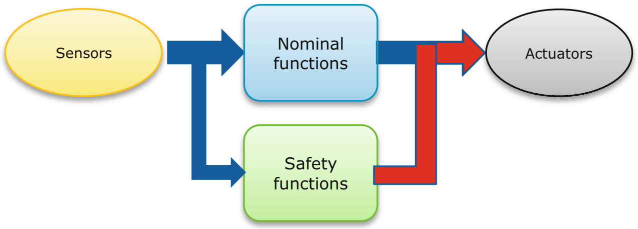


Fig. 3. Signal flow and functions

So-called "safety functions"[2] monitor the behavior of the control system. In case of E/E-fault occurrence, it is the duty of the safety functions to undertake all necessary actions (e.g. overwrite the setpoints from Level 1) to prevent the users[3] from unreasonable harm. The Level 2 functions are therefore instances that are working in the background and are typically not visible for the users. As representative examples for this paper, three different control systems have been tested and calibrated to prove the applicability of the proposed activities: a hybrid control system (HCU) and a transmission control system (TCU) consisting two variants (for conventional powertrains and for hybrid powertrains). The functional requirements for Level 2 were clustered in different safety mechanism and safety related functions (see Table 1). Although Level

[1] According to the E-Gas concept [5] these functions are often referred to as Level 1 functions.

[2] According to the E-Gas concept these functions are often referred to as Level 2 functions.

[3] Users in that sense covers all humans that can be affected by the vehicle or its functions (e.g. driver, passengers, maintenance staff or pedestrians).

2 is strongly dependent on Level 1, details on Level 1 are not explicitly mentioned in here. Testing these Level 2 functions is the challenge that should be addressed in the subsequent sections.

Table 1. Safety requirements and safety related functions

	HCU	TCU conventional	TCU hybrid
# of safety mechanisms/safety related functions	38	44	44
# of safety requirements (functional requirements)	361	412	420

3.1 Fault-Injection Testing vs. Robustness Testing

Before the testing of the safety function can start, it is required to have an aligned function calibration between Level 1 and Level 2. Whenever the Level 1 calibration changes, its relation to the safety function needs to be highlighted as the calibration changes might have an impact on the safety test results. Unintended interferences can appear e.g.:

- Level 2 reactions are too harsh compared to Level 1 reactions (e.g. due to parameter threshold violations)
- Level 2 reactions are not sufficiently debounced such that already triggered Level 1/diagnosis reactions cannot become effective

An aligned calibration data set can be derived by means of mathematical rules. These rules shall describe the relationship between the Level 1 function calibration data and the Level 2 function calibration data. The rules need to be applied to every dataset before the calibration/testing procedure starts (see Fig. 4) in order to make the safety responsible aware about potential interfering control actions.

To verify a software-intensive safety-related system, two testing approaches are necessary for functional testing which complement each other:

- Robustness testing
- Fault-injection testing

Robustness testing is a testing approach which can be used to verify if the system does not intervene when it is exposed to different operating scenarios which do not explicitly include any safety-critical situation (e.g. Is no safety reaction triggered in case of regular temperature conditions?). With respect to the Level 1 calibration dataset, this means that the Level 2 calibration parameter values must be chosen “sufficiently high” to not trigger unintended safety reactions. Considering only this aspect will lead to a robust, but not necessarily safe control system.

Fault-injection testing is a testing approach which can be used to verify if the system intervenes as specified in presence of a potentially safety-critical situation (e.g. Is a safety reaction triggered in case of overtemperature?). Together with the safety validation, it is the task of the fault-injection testing to ensure that the safety reaction takes place when needed (i.e. the Level 2 calibration parameter are chosen "sufficiently low"). Only if a system successfully passes

- a robustness testing campaign and
- a fault-injection testing campaign,

then it is adequately safe and robust with respect to safety-critical situations.

The workflow on how to combine the two approaches from the calibration perspective can be seen in Fig. 4. A detailed description of the efforts to be spent for each test approach and what challenges the engineers face with are explained in the subsequent sections.

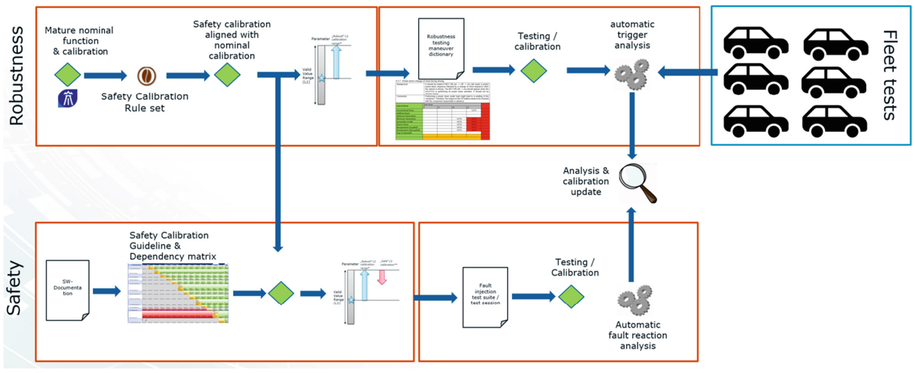

Robustness
Mature nominal function & calibration
Safety Calibration Rule set
Safety calibration aligned with nominal calibration
Robustness testing maneuver dictionary
Testing / calibration
automatic trigger analysis
Fleet tests
Analysis & calibration update
Safety
SW-Documentation
Safety Calibration Guideline & Dependency matrix
Fault injection test suite / test session
Testing / Calibration
Automatic fault reaction analysis

Fig. 4. Calibration workflow for fault-injection testing vs. robustness testing

3.2 Fault-Injection Testing

The expectation with respect to fault-injection testing is a dedicated safety reaction. For this purpose, the system is intentionally provoked by an injected fault. Typically, a fault is injected into the control system functions or into its related communication. By means of fault-injection, the complete fault propagation chain, shown in Fig. 5, shall be checked[4].

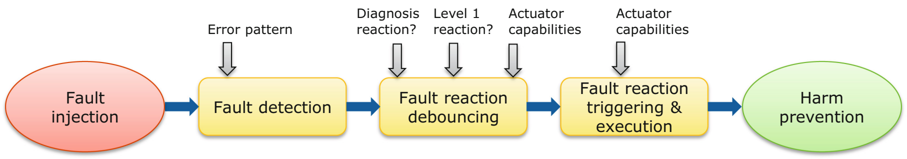


Fig. 5. Level 2 harm prevention strategy

[4] This implies that the fault reaction execution must be always switched on.

Starting points for the test activities are the input signals from the Level 1, the input signals from Level 2 and the functions defined for both levels[5]. All necessary information shall be available within the SW documentation and calibration guidelines (see Fig. 4). Together with the information about diagnosis functions, a formal test sequence (test session) can be derived. This sequence can either be used for manual testing or for automated testing. It is recommended to use an automated test approach to be able to perform the test loops in a time-efficient and highly reproducible way. To do so, the test specification is converted into a test automation sequence (test script) which can be executed e.g. on a Hardware in the loop system (HIL) (see Fig. 6) or on other test systems. It is recommended to define a test sequence is such a way (e.g. generic test description language) that it can easily be transferred between different test environments. The test sequences typically also contain automated check algorithms (pass-criteria[6]), that evaluate the effectiveness of the safety functions with respect to

- quantitative criteria (e.g. Is the reaction triggered when a threshold is exceeded?) and
- timing criteria (e.g. Is the reaction triggered sufficiently fast?).

When performing the test, a very important side-effect must be considered: As shown in Fig. 5, there might be an overlap between Level 1, Level 1 diagnosis and Level 2 with respect to the reaction that each system might trigger. When testing Level 2 functions, this overlap needs to be considered. Example: When Level 1 and Level 2 should trigger the same reactions, it is required to switch-off Level 1 reactions in order to observe just Level 2 reactions.

Due to the need of sharing test resources over different projects, the setup of modular test environments is recommended. To take this into account, a multi-HIL environment is of special interest. Multi-HIL in that sense means that different control systems e.g. HCU, TCU hybrid and TCU conventional can be tested on one HIL test bench. A schematic overview about such a HIL system can be derived from Fig. 6.

For fault-injection testing it is typically required to perform communication tests, where the communication between different control systems is corrupted (e.g. data corruption, message timeout [2]). Furthermore, fault-injection on hardwired interfaces (i.e. sensor lines) are performed. Such error patterns can contain signal drifts, short-to-ground, short-to-supply, … [2] and are typically injected by break-out-boxes. An addition to data corruption/manipulation outside of the control unit, also a fault-injection inside of the control unit (e.g. by manipulating calibration parameters) are applied.

[5] The overlap caused by calibration can be included in MOCA.

[6] In case of a non-successful evaluation of the safety functions, a root-cause analysis followed by rework-steps need to be performed. The same might hold true when calibration changes or functional changes have been applied to the control system.

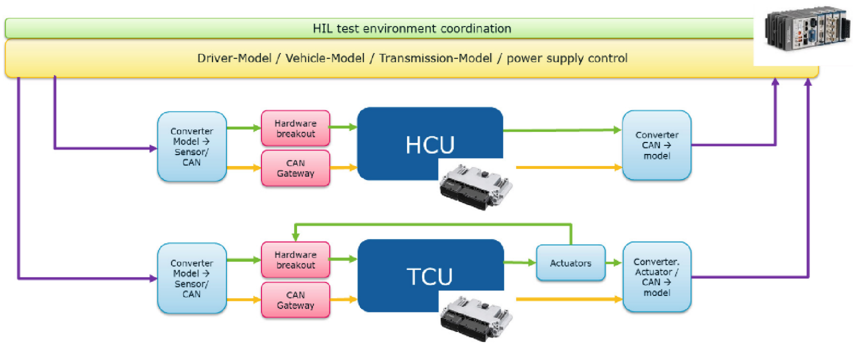

HIL test environment coordination
Driver-Model / Vehicle-Model / Transmission-Model / power supply control
Converter Model → Sensor/ CAN
Hardware breakout
CAN Gateway
HCU
Converter CAN → model
Actuators
TCU
Converter. Actuator / CAN → model

Fig. 6. HCU-TCU-Multi-HIL architecture (simplified)

When the fault-injection tests have successfully passed the HIL environment tests, then selected fault-injection tests are typically performed in the vehicle[7]. This is required to counterprove the reactions under real world conditions. Furthermore, the in-vehicle tests are required to check the functions and their calibration with respect to the validation targets.

3.3 Robustness Testing

The expectation with respect to robustness testing is that the system under test does not trigger any safety reactions when it is exposed to uncritical scenarios or maneuvers. It is a passive method were, in contrast to fault-injection testing, no "active" fault reaction is provoked.

The safety functions must observe the entire system consists of e.g. interfaces, Level 1 functions, sensors and actuators under all possible driving conditions and trigger safety reactions if required. A safety reaction without presence of E/E-faults would lead to customer dissatisfaction. Therefore, robustness testing tries to identify weak points in the safety functions and calibration by covering as much as possible combinations of vehicle driving scenarios and environmental and disturbance factors (e.g. temperature, humidity, EM radiation). Examples for aspects that characterize vehicle driving scenarios are mentioned in Fig. 7.

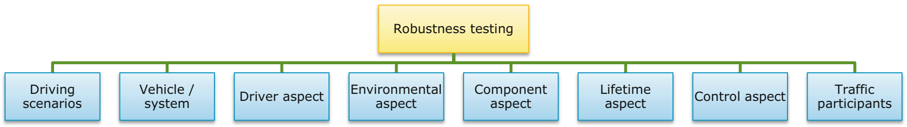


Fig. 7. Impact factors on driving scenarios over a vehicle lifetime (examples)

[7] Having a clever testing strategy in mind and having the test cases specified in a generic (ideally machine readable) manner would support to transfer of the test cases between the test environments.

Combining all possible influencing factors and variations to a set of driving scenarios would lead to an incredible number of test scenarios. As testing all scenarios is unrealistic with respect to development time and development costs, a more structured approach is required. In many projects, robustness testing is performed by a hands-on approach, where the interaction between driver and vehicle was selected to be the impact factor number one.

A robustness testing maneuver dictionary by considering the

- developed vehicle features (electric driving, creeping, snow-rock, …),
- potential driving scenarios (forward/rearward driving, speed range) and
- potential driver inputs (gear lever change, accelerator pedal input, …)

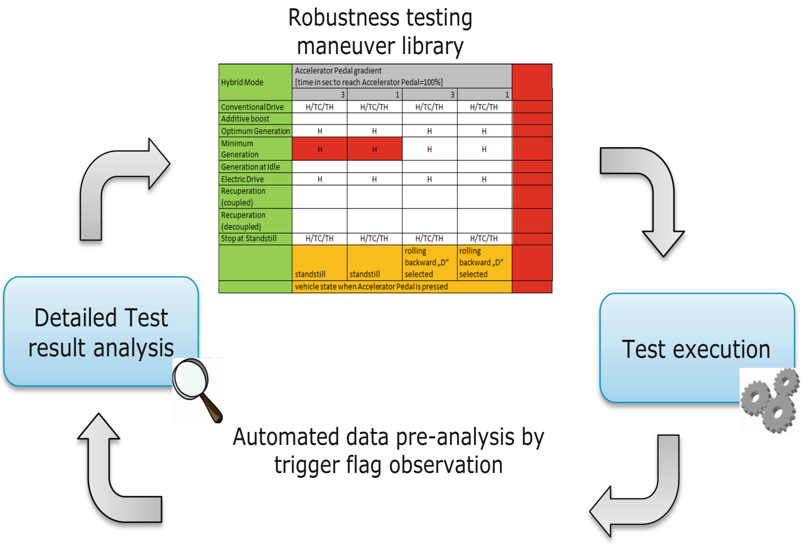
was created (see Fig. 4).

This dictionary contains several hundred scenarios that can occur during a vehicle lifetime. They are the starting point for the robustness test execution. The test execution is typically started in the vehicle (i.e. on a test track)[8]. When the functions have a certain maturity i.e. no unintended safety reactions are triggered, then the defined scenarios are backed-up by real world drives (e.g. test trips driven in real traffic situations). A robustness testing maneuver library is generated and can be tested by means of a complete vehicle fleet. The fleet test data is collected via a data logger and transferred via a cloud solution into a back-end where a further data processing (e.g. big data) is performed.

Figure 8 shows the steps that are required to achieve robust safety functions. Due to the fact, that all the functions are observed at the same time, and the scenarios cannot be exactly repeated, the amount of data that must be logged and analyzed increases dramatically compared to fault-injection testing. Efficient checking of each function manually would not be possible. To overcome this situation, an automated pre-analysis mechanism is applied. As a result of the pre-analysis procedure, a test report is generated. This report provides the safety experts a summary about all potential safety triggers including the information of trigger time and source[9]. Instead of analyzing all the data manually, the analyzing team just needs to focus on the trigger flag in the time window in which the safety intervention occurred.

[8] To perform an efficient robustness testing, it is proposed to “guide” the test driver through the test maneuvers. This is done by a human-machine-interface that gives the driver an explanation of the intended test before the test shall be performed. After the test has finished, a check routine is automatically triggered. The result of this check routine will immediately inform the driver if the test maneuver was successful or the test must be repeated.

[9] Please note, that depending on the test progress the “fault reactions triggering & execution” (see Fig. 5) might be switched off. The tester has no immediate feedback about potential safety triggers and its related fault reaction. As some safety triggers are only healed by an ignition cycle, a visual and audible HMI notification is provided to the tester.


Robustness testing maneuver library
Test execution
Automated data pre-analysis by trigger flag observation
Detailed Test result analysis

Hybrid Mode	Accelerator Pedal gradient [time in sec to reach Accelerator Pedal=100%]			
	3	1	3	1
Conventional Drive	H/TC/TH	H/TC/TH	H/TC/TH	H/TC/TH
Additive boost				
Optimum Generation	H	H	H	H
Minimum Generation	H	H	H	H
Generation at Idle				
Electric Drive	H	H	H	H
Recuperation (coupled)				
Recuperation (decoupled)				
Stop at Standstill	H/TC/TH	H/TC/TH	H/TC/TH	H/TC/TH
	standstill	standstill	rolling backward „D“ selected	rolling backward „D“ selected
	vehicle state when Accelerator Pedal is pressed			

Fig. 8. Robustness-test cycle

When having robust safety functions under nominal conditions, it is recommended to increase the overall function and calibration maturity. This can be done by introducing further impacting factors (see Fig. 7). One possible approach to increase the maturity is the repetition of the tests under real-world conditions (e.g. by performing tests in different vehicles or with different drivers). As this possibility is time-consuming and expensive, another approach is of preferable interest (see Fig. 9). The preferred approach reduces the previously mentioned disadvantages, as a “semi-virtual” test environment is utilized. Semi-virtual in that sense means, that vehicle measurement data is introduced on a HIL as target values for the virtual environment. E.g. the measured vehicle speed is trusted to be the target speed for a HIL driver model. System parameters (e.g. lifetime aspects, environmental aspects, …) can be varied and the advantages provided by a HIL (e.g. 24/7-utilization) can get a chance.

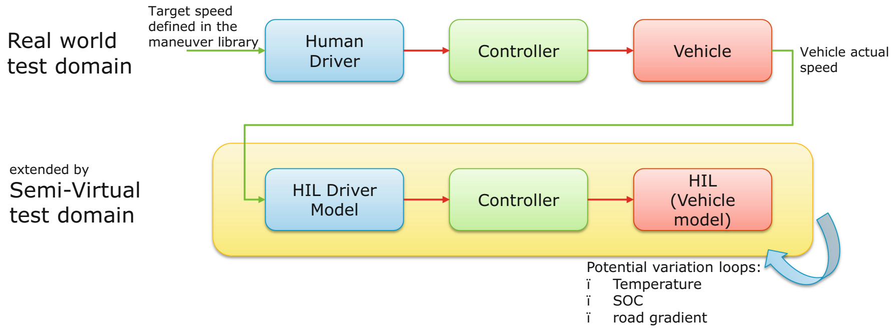


Fig. 9. Increase of function maturity by a semi-virtual test approach

Robustness testing is typically performed in parallel to the fault-injection testing or in a sequence like shown in Fig. 10. Special attention must be put on the fact that robustness testing and fault-injection testing have contradicting goals with respect to calibration. While fault-injection testing typically leads to a reduction of the safety margin, robustness testing typically tends to enlarge safety margins.

3.4 System Item Integration and Testing

In the previous sections, the focus was purely put on the test methods used within the safety function development. Remembering Fig. 3, it is obvious that the nominal functions and its related diagnosis functions must be considered in the safety development as well. Experience has shown that the functional growth and the reconcilement of all functions are done in a sequence like shown in Fig. 10.

Starting point for all required activities are released Level 1 functions[10]. After performing initial integration steps, the initial robustness is tested (second part in Fig. 10). As the safety functions are introduced the first time, it is required to minimize the interaction between Level 1 and Level 2. This is ensured by deactivating the diagnosis reaction in the Level 1 software. At a later stage, when the initial robustness tests are passed, the diagnosis reactions are switched on. Safety functions and diagnosis functions can now be harmonized.

As shown in Fig. 10 (third part), the safety test development is continued by performing fault-injection testing. Following the fault propagation chain mentioned in Fig. 5, the fault detection needs to be checked first. When the detection mechanisms work as expected, then the safety reactions are switched on. From that time onwards, the whole propagation chain can be tested and calibrated.

In a final step (fourth part in Fig. 10), it is required to activate the all currently switched-off functions (e.g. diagnosis) to check the overall interaction of Level 1 and Level 2. It is recommended to perform a final loop of robustness tests and fault-injection tests to conclude the item integration and testing[11].

[10] In minor cases it is also possible to start from a sufficiently mature Level 1 function. Level 1 function calibration and Level 2 function testing may go in parallel but bear the risk of function mismatch leading to additional development efforts.

[11] Between the single test steps shown in Fig. 10 it might be required to change the implemented functionality (e.g. bug fix or calibration update). Whenever this takes place, it might be that parts of the procedure must be repeated.

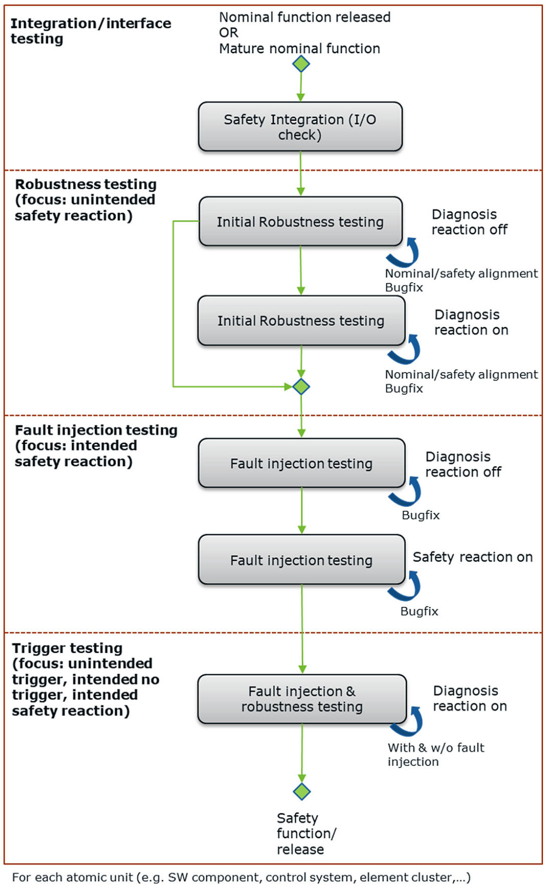

Integration/interface testing
Nominal function released
OR
Mature nominal function
Safety Integration (I/O check)
Robustness testing (focus: unintended safety reaction)
Initial Robustness testing
Diagnosis reaction off
Nominal/safety alignment
Bugfix
Initial Robustness testing
Diagnosis reaction on
Nominal/safety alignment
Bugfix
Fault injection testing (focus: intended safety reaction)
Fault injection testing
Diagnosis reaction off
Bugfix
Fault injection testing
Safety reaction on
Bugfix
Trigger testing (focus: unintended trigger, intended no trigger, intended safety reaction)
Fault injection & robustness testing
Diagnosis reaction on
With & w/o fault injection
Safety function/ release
For each atomic unit (e.g. SW component, control system, element cluster,...)

Fig. 10. Safety function testing and calibration workflow (process view)

4 Future Work

Although some proposals for future improvement have been stated in the paper, there are still unexploited potentials. It is planned to further increase the test depth and the test efficiency over the whole lifecycle.

Based on the recommendation of the ISO26262:2018 it is intended to enforce semi-formal and formal methods/approaches. One example is the application of additional formal aspects related to requirements engineering and test specification. Requirements are typically written in unconstrained natural language, which makes them prone to problems like for example ambiguity, complexity and vagueness [6]. Especially the specification of requirements/test sequences in a controlled natural language as e.g. proposed in [6] sounds promising for further automation.

First tests in introducing patterns already in an early development phase have shown very promising results (i.e. consistency checks, …). The gained results and the available lessons learnt will be used to extend and improve the test specification, test automation, test execution and test analysis activities.

5 Conclusions

This paper shows a systematical and “proven in use” approach for “System Item Integration and Testing” and “Safety Validation”. It highlights and describes the necessary steps to perform a proper item integration in context of functional testing. A special focus is on the interaction between the two test approaches of fault-insertion and robustness testing. Details of both test methods and its relationship to different test environments are shown. Furthermore, it illustrates that the development of a safe product is not a straight-forward task. Especially the verification and validation activities are of big importance to bring a safe product on the market. Finally, to further improve the verification and validation activities an outlook on promising next steps is given.

References

1. International Electrotechnical Commission: IEC 61508 Edition 2.0: Functional safety of electrical/electronic/programmable electronic safety-related systems (2010)
2. International Standardization Organization: ISO 26262: Part 1-12: Road vehicles – functional safety (2018)
3. Wambera, T., Macher, G., Frohner, B.: Prozesssteuerung und domänenspezifische Dokumentation nach ISO26262 und ISO25119 während der Entwicklung und Integration von sicherheitsrelevanten Systemen, Diagnose in mechatronischen Fahrzeugsystemen XIII: Neue Verfahren für Test, Prüfung und Diagnose von E/E-Systemen im Kfz. TUDpress, Dresden (2019)
4. Pries-Heje, J., Johansen, J.: The SPI Manifesto (2009). https://2020.eurospi.net/images/eurospi/DownloadCenter/spi manifesto.pdf
5. Arbeitskreis EGAS: Standardisiertes E-Gas Überwachungskonzept für Benzin und Diesel Motorensteuerungen. Arbeitskreis EGAS, Version 6.0
6. Mavin, A., Wilkinson, P., Harwood, A., Novak, M.: Easy approach to requirements syntax (EARS). In: 2009 17th IEEE International Requirements Engineering Conference (RE 2009), pp. 317–322. IEEE, Atlanta (2009)
7. Holtmann, J., Meyer, J., von Detten, M.: Automatic validation and correction of formalized, textual requirements. In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, pp. 486–495. IEEE, Berlin (2011)