Requirement Decomposition and Testability in Development of Safety-Critical Automotive Components[*,**]

Viacheslav Izosimov, Urban Ingelsson, and Andreas Wallin

EIS by Semcon AB, Sweden
{viacheslav.izosimov,urban.ingelsson,
andreas.wallin}@eis.semcon.com

Abstract. [12]ISO26262 is a recently approved standard for functional safety in road vehicles. It provides guidelines on minimization of unreasonable safety risks during development of embedded systems in road vehicles. However, the development process specified in ISO26262 involves a number of steps that will require changing traditional and well established development processes. In a transition phase, however, due to lack of tool support, the steps may be performed manually, increasing the risk for delays and increased cost. This paper describes a case study in which we have successfully worked with traceability and testability of functional safety requirements, as well as safety requirements assigned to a testing tool that automates integration and verification steps, leading to standard-compliant tool qualification. Our tool qualification method employs fault injection as a validation method to increase confidence in the tool. Our case study will help to avoid many of the new pitfalls that can arise when attempting to realize standard-compliant development.

1 Introduction

Industry and academia struggle to improve safety of road vehicles. The innovations often employ embedded systems. However, malfunctions in safety-critical embedded systems may lead to new hazards (potential sources of harm). To reduce the risk of such malfunctions, safety-critical embedded systems must be developed according to a safety standard. Recently a standard for functional safety, IEC61508, was adapted to the context of road vehicles resulting in ISO26262 [1], which addresses development of safety-critical electronic systems (Items). Development steps and processes are specified according to five Automotive Safety Integrity Levels (ASILs), namely Quality Management (QM) and ASIL A-D. The ASIL for an Item is determined by

[*] This work has resulted from FUSS, a subproject of DFEA2020, partially sponsored by the FFI council of VINNOVA (Swedish Research Agency).

[**] The authors thank Erik Hesslow, an ISO26262 safety expert from Mecel AB (partner in the FUSS/DFEA2020 project), for reviewing the work and providing valuable comments.

considering the severity and probability for each hazard, as well as a driver's ability to compensate (controllability). For high ASIL items, the standard requires stringent measures for risk minimization. In contrast to IEC61508, besides many other aspects, ISO26262 imposes qualification requirements on software tools used in the development process, which also includes verification and validation tools. While tools exist, they may not have been qualified or developed considering safety requirements. Consequently, to be ISO26262-compliant, existing tools must be qualified, and in each future version, re-qualified. Similar to IEC 61508, ISO26262 allows decomposition of high ASIL safety requirements, using two same-or-lower ASIL requirements and redundancy, monitoring or other safety-enhancing concept. Since development to a lower ASIL typically requires less effort, decomposition is an attractive possibility. However, the decomposition must be implemented by independent components and affects the system architecture. To demonstrate fulfillment of the original requirements, there shall be traceability to and from the decomposed requirements.

As seen from above, ISO26262-compliant development include specification of safety requirement (including determination of ASIL), decomposition of safety requirements, requirement traceability and testability, qualification of software tools, verification and validation. This paper provides an example of how these steps can be performed. The aim is to help minimize pitfalls in transition to ISO26262.

The next section reviews prior work. Section 3 presents requirements elicitation and traceability. Section 4 discusses testability, leading up to Section 5 which is about testing tool qualification. Section 6 presents a verification and validation strategy. These concepts are illustrated in a case study in Section 7.

2 Prior Work

Previous publications on ISO26262 include introductions to the standard [2] [3] [4] [5], guides to successful application [4], experience reports [5], studies on the impact on Item development [6], considerations regarding the development process and assessment [3] [7] and adapting model-based development workflows to the standard [8]. Dittel and Aryus [2] pointed out the need for support tools and methods. Hillenbrand, et al. [6] discussed impact on the electric and electronic architecture, as well as management of safety requirements. They found challenges, time-consuming activities, lack of support tools and proven workflows [8].

Support tools for ISO26262-compliant development are considered in [9] [10] [11] [12]. Makartetskiy, Pozza and Sisto [9] review two tools, Medini and Edona, for system level modeling, handling the documents and checks against the standard regulations. They stress that to bring a shared view of safety among companies, both a standard and tools are required. Hillenbrand et al. [10] provide an FMEA tool with features to support work with ISO26262. Schubotz [11] address the gap between the standard and companies' internal development processes by a concept approach to plan, link, track and evaluate standard-required activities with documentation. Palin, Ward, Habli and Rivett [12] argue that a safety case consisting of collected work products, as ISO26262 allows, lacks an explicit argumentation for safety. They present a template for a proper safety case using goal structuring notation.

Qualification methods for software tools used in development are addressed in [13] [14]. Conrad, Munier and Rauch [13] present a reference development workflow using model-based design, with checks in every development step, comparing requirements and the model and comparing test results and the model. This way, tool confidence is achieved by high tool error detection probability. In [13] the tools are qualified for such use that strictly follows the reference development workflow. The reference workflow approach to tool qualification is criticized by Hillebrand, et al. [14], since it is tailored to specific tools and creates a dependency on the tool vendor. While it is good practice to keep the same tool version throughout a development project, various projects use different tool versions. This can be a source for confusion. A "tool" may be a flow consisting of several tools and each tool in the tool flow may have to undergo qualification. In [14] tool classification is addressed to avoid unnecessary effort in tool qualification.

Robinson-Mallett and Heers [15] report that hardware-in-the-loop (HIL) test platforms require special consideration, and the model-based approaches to tool qualification do not apply. HIL test platforms provide a test environment that closely resembles the intended operation environment of the Item and can be more complex than the sum of electronic components in a car. Consequently, qualification of a HIL platform is a challenge. In our previous work in [16] and [17], a testing tool qualification method for HIL platforms is presented to reduce the qualification effort. The method includes a monitor and fault injection. Our work in [16] and [17] focus on development of a semi-automatic qualification process for the HIL tool, while we do not consider traceability and testability of the Item requirements and within the HIL tool.

The papers listed above have identified the need for a best practice and the need to develop and qualify tools. Previous papers on ISO 26262 have not discussed requirements traceability of safety-critical systems in the context of decomposition, nor for verification and validation. For non-safety-critical complex computer-based systems, however, Arkley and Riddle [18] discuss requirement traceability, motivating the need for a traceable development contract. Further, in the context of aerospace industry, Andersen and Romanski [19] discuss development of safety-critical avionics software, including verification, validation and assessment, and emphasize importance of requirement traceability. Neither of the previous papers, however, has addressed propagation of safety requirements into the tool qualification. To address tool qualification, requirements traceability, verification and validation in the context of safety-critical systems and ISO 26262, this paper provides an example of how such tasks can be performed, illustrated by a case study.

3 Safety Requirement Elicitation and Traceability

To get an overview of the activities that are involved in elicitation and traceability of safety requirements, Fig. 1 shows the safety lifecycle, i.e. the safety-related steps of a development project that follows ISO26262. Each step has a set of work products, as content for design documents and item documentation. Item definition is both the first step and the first work product, in which the concept item is described, before development starts. Only product and project requirements are gathered here. Functional safety requirements, i.e. requirements that must be fulfilled to achieve minimization of unreasonable safety risks, are identified in the subsequent hazard analysis and risk assessment step. Hazards are identified and categorized leading to an ASIL assignment and a set of safety goals. A safety goal is an abstract, top-level safety requirement, which is proposed to overcome the hazardous situation that can arise from malfunctioning of the Item, mitigating the risk that this situation brings. To fulfill the safety goals, more detailed safety requirements are defined, each with a corresponding ASIL. Thus, a functional safety concept is formed, consisting of all the safety requirements and the steps taken to ensure safety. The safety requirements govern all subsequent steps of the safety lifecycle. A typical problem in any large project is that an individual requirement does not explain the reasoning behind its formulation and so the importance of a safety requirement can be misunderstood. To clarify relations between requirements and their reasons, *requirement traceability* is ensured by linking each requirement to safety goals, corresponding tests, design decisions, etc.

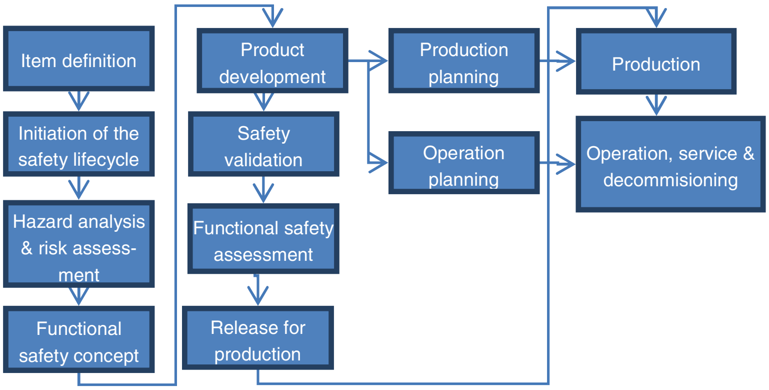
Item definition
Initiation of the safety lifecycle
Hazard analysis & risk assess-ment
Functional safety concept
Product development
Safety validation
Functional safety assessment
Release for production
Production planning
Operation planning
Production
Operation, service & decommisioning

Fig. 1. Safety lifecycle

Requirement traceability is illustrated in Fig. 2, where some of the most relevant ISO 26262 work products are overlaid on a V-type development process. As shown with lines between the work products on the left hand side of the V, requirements are specified in several steps, including the safety goals, the functional safety concept, the technical safety requirements and the specific hardware and software requirements. The included work products show how requirements must be traceable in work products for test, integration, verification, qualification, assessment and validation. Traceability of requirements on independence after decomposition, and between different ASILs, must be also addressed in all the steps, including the safety case.

As can be seen from Fig. 2, lower abstraction levels contain more detailed requirements. Different shades of the work products describe different sets of requirements, namely safety goals, functional safety requirements, technical safety requirements, software requirements and hardware requirements. The requirements in a given set are designed to fulfill the requirements in the set above in a more specific way. It should be noted that requirement traceability must correctly describe how a set of requirements fulfill the set above. Such traceability of safety requirements can practically be implemented by tabulating the relations between requirements in each work product. Such a table should detail the name of the requirement, the name of requirements with "fulfills" or "fulfilled by" relations and the names of all other related requirements. An example is given in Table 1.

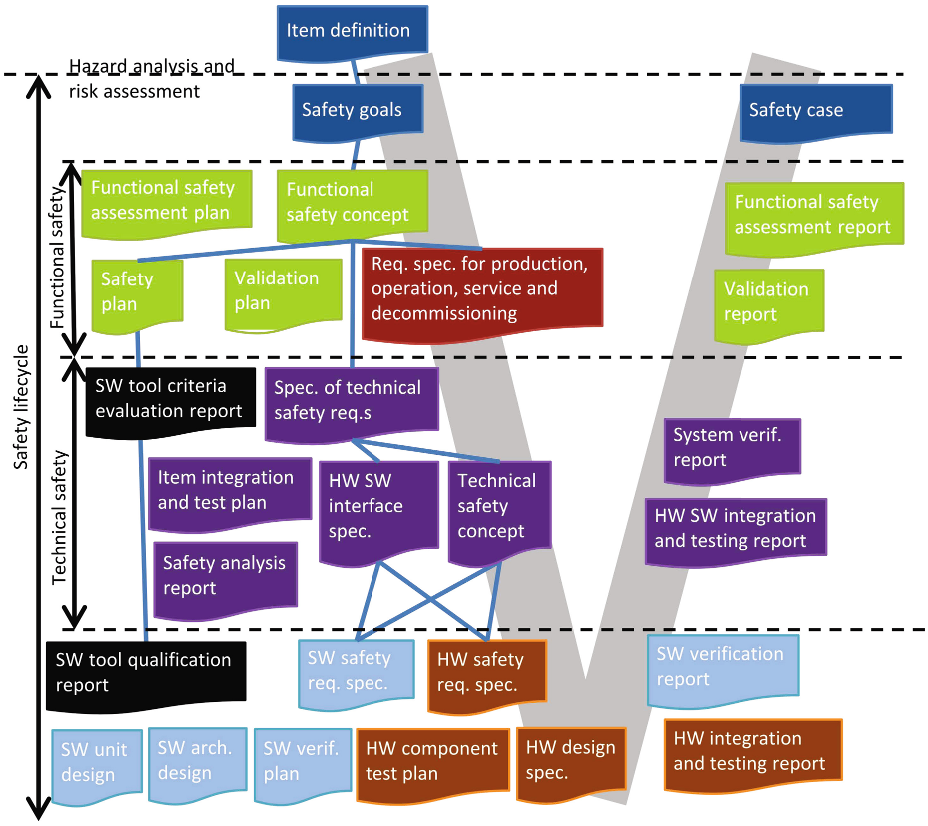
Item definition
Hazard analysis and risk assessment
Safety goals
Safety case
Safety lifecycle
Functional safety
Functional safety assessment plan
Functional safety concept
Functional safety assessment report
Safety plan
Validation plan
Req. spec. for production, operation, service and decommissioning
Validation report
Technical safety
SW tool criteria evaluation report
Spec. of technical safety req.s
System verif. report
Item integration and test plan
HW SW interface spec.
Technical safety concept
HW SW integration and testing report
Safety analysis report
SW tool qualification report
SW safety req. spec.
HW safety req. spec.
SW verification report
SW unit design
SW arch. design
SW verif. plan
HW component test plan
HW design spec.
HW integration and testing report

Fig. 2. Work products and requirement relations

Table 1. Relations between TSR42 and other requirements (often represented by "links" between requirements in requirement management tools)

Requirement ID	Fulfills	Fulfilled by	Other related
TSR42	FSR17	HWSR71, SWSR50	TTR3

Table 1 shows a technical safety requirement TSR42, which is designed to fulfill the functional safety requirement FSR17 together with other requirements. Similarly, hardware safety requirement HWSR71 and software safety requirement SWSR50 are designed to fulfill TSR42. Further, TSR42 is related to TTR3, a requirement assigned to a testing tool. To span all the sets of requirements, there should be similar tables to relate HWSR71 to hardware components, SWSR50 to a part of the software design, and FSR17 to a safety goal. Following these relations in either direction helps in understanding the requirements and the Item.

The above requirements traceability concept is common practice in all mature development projects. ISO26262 requires adaption of this common practice to decomposition of requirements and tool qualification. Furthermore, requirement traceability is a prerequisite for requirement testability, which is discussed next.

4 Testability

The claim that design and implementation fulfill the requirements shall be verified. *Verification* is the task to determine completeness and correct specification of requirements as well as correctness of the implementation that is to fulfill the requirements. Verification constitutes the right hand side of Fig. 2 and is performed for all integration steps of the system design including implementation in software and hardware. To be verified, the requirements should be testable. To ensure testability, a semi-formal representation that is compatible with a definition of testability is utilized. We present two representations to illustrate aspects to testability.

For the first representation, we define a requirement *Ri* as a logical expression *Li*: <Object *X*> **shall** <Action *Y*> [applied to] <Subject *Z*>. The requirement is mapped onto Object *X* which performs Action *Y* onto Subject *Z*. Testability of *Ri* is a property of *Ri* that this logical expression *Li* can be verified. We suggest that, to fulfill testability, the requirement has to consist of the object, the action and the subject and the object, the action and the subject must be identifiable within and present in the system. With these conditions valid, the requirement can be verified, and is *testable*.

Consider following "good" and "bad" examples of safety requirements, some that fulfill, and some that do not fulfill the requirement pattern and, thus, shall be changed:

R1: We shall ensure presence of error correction codes (ECC) in the system for correction of single-event upsets (SEUs).
R2: The MCU (microcontroller) shall include a logical watchdog.

In R1, neither Object nor Subject is clear, only Action is present, i.e., ensuring presence of ECC codes, and the requirement is not testable. In R2, the elements are clearly identifiable and physically present in the system. Thus, this requirement is testable. However, this requirement will have to be detailed further to identify watchdog properties, relevant MCU software and monitoring strategy.

For the second representation of requirements, consider that although object, action and subjects are obligatory attributes of requirements, it is often important to identify *conditions* under which the requirements are applicable. R3 is an example requirement that is designed to prevent over-heating of a component.

R3: The MCU shall not enable a power supply to the central CPU if the ambient temperature is above 95°C.

In R3 there is an example of another important property of requirements, which is the presence of measurable quantitative parameters. These parameters will ensure operational intervals and applicability of requirements, i.e., as in R3, "above 95ºC". However, R3 is not easily refutable. The test that is necessary to check that the requirement is fulfilled will be boundless. Therefore, it is good practice to either formulate requirements such that they are easily refutable or give a set of appropriate measurement conditions for the test.

Requirement elicitation with respect to requirement testability and how it leads to testing tool qualification can be shown in several steps (see Fig. 2 for work products):

Define Safety Goals: Safety goals cannot be tested since they are usually very abstract. Note, however, that safety goals and functional safety requirements shall be *validated* by studying behavior of the whole system, to ensure that the correct system has been developed and potentially dangerous behavior successfully avoided.

Define Safety Requirements: Many functional safety requirements cannot be verified due to lack of technical details. In this step, however, it is usually clear which testing tools will be needed. Thus, selection and classification of testing tools can be done, resulting with an input to SW tool criteria evaluation reports.

Refine Safety Requirements: By considering system properties, decomposition of requirement is performed. Requirements are also evaluated on their feasibility by performing requirements reviews, design studies and testability assessments. This will result in a verification strategy, part of which will be adaptation of the test tool.

Detailed Safety Requirements: Verification is possible only for technical safety requirements, which are the most detailed safety requirements. In this step, it is necessary to derive test cases and clearly demonstrate requirement testability. Several iterations of requirement elicitation may be needed. Testing tool qualification is performed, resulting with input to SW tool qualification reports.

Implementation: Here, verification activities are fully executed on implementation releases with testing tools providing test reports for the respective requirements.

Safety Case: Test cases, test reports and tool qualification reports will provide inputs to the safety case, for demonstration of fulfillment of the requirements.

5 Testing Tool Aspects of Testability

Verification of safety requirements is usually done with help of a testing tool, to automate the verification process and increase its efficiency. A testing tool is used to verify the logical expression of a requirement (see Section 4) by applying test cases, generated or specifically provided for this requirement. Some testing tools, in particular hardware-in-the-loop test rigs, often need to be adapted for testing against safety requirements. In the following, we consider such a testing tool with regard to the requirement aspects. For the testing tool, we will have to specify functional safety requirements. Such specification includes *classification* and *qualification* of a testing tool. Classification will identify which measures are to be applied for ensuring correctness of the testing tool. The classification has three Tool Confidence Levels (TCLs) and depends on tool error detection (TD) capability and tool impact (TI). Tools that have a possibility of silent failures (TD3) with high impact to the Item (TI2) motivate qualification to the highest confidence level, TCL3. A "silent" malfunction in a testing tool used for an ASIL D Item can cause a test to miss detection of a fault in a component of a road vehicle.

Qualification will ensure correctness and applicability of the testing tool based on the classification. ISO26262 specifies qualification steps according to the TCL that is required from the tool. For example, when classification determines that tool malfunction can cause an ASIL C or ASIL D safety hazard and this malfunction is likely not to be detected, the standard recommends that the tool should be developed to the same ASIL according to a safety standard, followed by validation. Development and validation of the testing tool should complement each other to ensure that the risk of test escapes in the safety-critical component is minimized. Note also that if the tools are used for testing of decomposed requirements, i.e., ASIL B(D), the ASIL level of independence, in this case: ASIL D, shall be often considered as the ASIL level in qualification of these software tools.

The results of qualification of a testing tool and verification against safety requirements of the safety-critical automotive component will be reflected in a work product called the safety case (see Fig. 2), which will include arguments that safety is achieved using qualification and verification work products, including testing tool analysis report, testing tool qualification report, integration and verification plan, test cases (for the respective requirements) and respective test reports.

It should be noted that verification includes more than testing against requirements. A complete verification process includes activities such as fault injection experiments, tests in the operation environmental of the Item and EMC tests.

6 Verification and Validation

As mentioned in Section 5, the main document to describe the argumentation for item safety is the safety case, which includes content of the work packages that are required by ISO26262. A significant part of the safety case comes from work products of verification and validation activities. All these activities and work products become difficult to manage without a thought-through and proven strategy. Part of any such strategy is to automate as much as possible, use templates to ensure information quality, and to have tool support for the activities and management of the work products.

However, automation requires extra effort with respect to tool qualification. In [16], a testing tool qualification method was presented, with a monitor to detect testing tool malfunction and fault injection into the testing tool to evaluate the capability of the monitor to detect malfunctions. The method is semi-automatic and reduces the effort for tool qualification as is described in the following case study. To enable efficient ISO26262-compliant development, it is vital to gather such methods and tools.

7 Case Study

In this section, we provide an example where we apply the concepts discussed in the previous sections, in particular decomposition, traceability and testability of requirements, as well as testing tool qualification and fault injection based verification.

7.1 ASIL C Windshield Wiper

Consider a car's windshield wiper and washer liquid spray. When the washer liquid spray is activated, the windshield wiper is also activated for a short duration.

Two failure modes of the windshield wiper controller may cause the driver's view to be obscured by washer fluid, by (1) failure of the windshield wiper or by (2) failure of the washer liquid spray. Controller failure can impact a common driving scenario, while driving at high speed on a curvy road, resulting in the highest probability, E4. The highest severity, S3, applies, since the result may be that the car departs from the road at high speed with risk of critical injury. The controllability is modest, C2, since an obscured view is comparable to loss of headlights at night, which is categorized as C2 in [1] (Part 3, Table B-4). Consequently, the hazard corresponds to ASIL C, the second highest ASIL ([1] Part 3, Table 4).

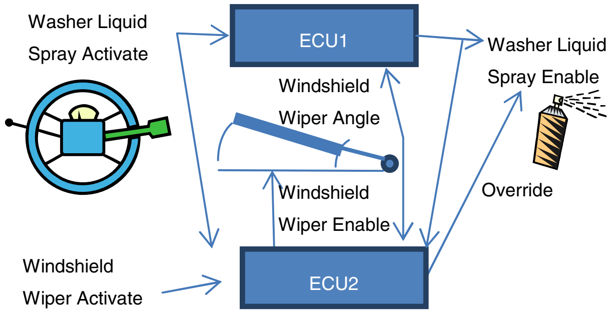

Fig. 3. Overview of windshield wiper

We formulate a safety goal **SG1**: "A malfunction should not obscure the drivers view with washer liquid". For SG1, we formulate two functional safety requirements, FSR1 and FSR2, to enforce a safe state "washer liquid spray disabled" upon controller failure. The two requirements correspond to the two possible failure modes.

FSR1: The controller should not spray washer liquid if the windshield wiper fails.
FSR2: The controller should not spray washer liquid for an extended duration.

We found a decomposition to fulfill both FSR1 and FSR2. An overview is given in Fig. 3. The ECUs perform mutual checking of each other's operation as is described by the technical safety requirements TSR1.1 and TSR2.1.

TSR1.1: ECU1 shall disable the washer liquid spray if the windshield wiper angle does not change.
TSR2.1: ECU2 shall override the washer liquid spray if the washer liquid spray is enabled for >1s.

ISO26262 allows decomposition from an ASIL C requirement to two requirements with ASIL A(C) and ASIL B(C) respectively, if they are independent, e.g. correspond to independent ECUs. ECU1 is controlling the washer liquid spray based on the driver's activation, while monitoring the windshield wiper angle. Thus ECU1 is to fulfill TSR1.1. ECU2 fulfills TSR2.1 and is responsible for controlling the windshield wiper based on the driver's activation and sensor input of the windshield wiper angle. ECU2 also monitors the washer liquid spray enable signal from ECU1 such that it can override that signal if necessary. We choose to assign ASIL B(C) to ECU2 and ASIL A(C) to ECU1 since ECU2 controls the windshield wipers. A malfunction of the windshield wipers can potentially lead to ASIL B hazards. Take, for example, a scenario in which the windshield is suddenly splashed with dirt which has been stirred up by another vehicle on a wet and dirty road. Visibility is suddenly reduced. Malfunction of the wipers in this situation will not allow cleaning of the windshield. Although the situation is fairly controllable (C2), the probability of this situation is second highest (E3) and the vehicle may drive into meeting traffic leading to high severity (S3) if the driver loses control. Thus, ASIL B should be assigned ([1] Part 3, Table 4).

The traceability of these requirements across the decomposition is implemented in Table 2 as described in Section 3. Testability is achieved by representing the technical safety requirements according to a semi-formal pattern (see Section 4) and by using quantitatively measurable parameters. A testing tool is required, as is discussed next.

Table 2. Requirement relations

Requirement ID	Fulfills	Fulfilled by	Other related
SG1	n/a	FSR1, FSR2	n/a
FSR1	SG1	TSR1.1	n/a
FSR2	SG1	TSR2.1	n/a
TSR1.1	FSR1	HWSRxx, SWSRyy	TTR1
TSR2.1	FSR2	HWSRww, SWSRzz	TTR2

7.2 Testing Tool Qualification

The considered testing tool consists of software and a Hardware-In-the-Loop (HIL) platform, which is to be qualified to TCL3, as discussed in Section 5.

ISO26262 recommends that tools with TCL3 are developed according to a standard for safety-critical systems and then validated (see Section 5). Even though the testing tool is not a component of a road vehicle, we develop the testing tool according to ISO26262 as an Item. The testing tool has the same ASIL as the Item with the highest ASIL that is to be tested, and we want to be able to test ASIL D items.

The testing tool is intended to be used during development, to get prototypes certified for use in road vehicles. As a prototype is developed, new features and attributes are added. This type of testing tools are often developed together with the prototype since the set of signals to measure and the evaluation criteria are not known on beforehand. Consequently, frequent changes to the testing tool can be anticipated. For each change to the testing tool, re-qualification to ASIL D is required. However, the effort involved in re-qualification of the testing tool to TCL 3, by management of changes to an ASIL D Item, can be a bottleneck for the development process. Consequently, we sought an appropriate decomposition to reduce re-qualification effort.

We added a monitor to the testing tool, such that the monitor is developed to ASIL D(D) and the testing tool to QM(D). The key idea behind this decomposition is that the monitor ensures detection of testing tool failures, bringing the tool error detection to TD1. This leads to the lowest required tool confidence level TCL1, for which less qualification effort is required. While the testing tool goes through frequent changes with re-qualification corresponding to TCL1, the monitor is not changed so often. Re-qualification to TCL3 through change management of an ASIL D Item is not required very often and there is less effort when the testing tool is changed.

We use fault injection experiments to semi-automatically perform verification and validation on the monitor [16]. In these experiments, we inject faults into the testing tool and thereby measure the monitor's ability to detect unexpected behavior in the testing tool. Through these experiments we identify three cases. The first case corresponds to discovering a "bug" in the testing tool. In this case, the decision about changing the monitor is deferred until the "bug" is corrected. In the second case, it is discovered that the monitor is insufficient and requires a change and a change management to ASIL D(D) is performed, followed by further fault injection experiments. In this case, the fault injection experiments must be adjusted. In [16] we describe a semi-automatic procedure for adjusting the fault injection experiments. In the third case, the monitor is able to detect all injected faults and no change to the monitor is required. The relative frequency of the three cases depends on the type of testing tool changes. We expect that the third case, which requires no changes to the monitor, will be common enough to motivate the decomposition by its reduction in effort.

7.3 Case Study Summary

In the case study, we have seen two different applications of requirement decomposition, explicit requirement traceability and thorough management of requirement testability including testing tool qualification. Furthermore, we believe that the fault injection experiments applied to verify the testing tool monitor can be adapted also to other software components and tools as an appropriate and time-saving verification method.

8 Conclusion

This paper addresses development of safety-critical embedded systems for use in road vehicles according to ISO26262. Since the standard is new and introduces development steps such as requirement decomposition and software tool qualification, we have argued that this can lead to many manual steps and consequential pitfalls.

For example, software tool qualification can become a bottleneck in the development process. To mitigate such pitfalls we have reviewed the important concepts requirement decomposition, traceability, testability, verification and validation. We have showed application of the concepts in a case study involving two requirement decompositions, testing tool qualification using a monitor and fault injection experiments. The chosen approach will increase efficiency of the development process of Items with high ASIL levels, avoiding unnecessary bottlenecks and potential pitfalls that might lead to hard-to-solve problems and compromise safety.

References

1. ISO, ISO 26262:2011 Functional safety - road vehicles, ISO (2011)
2. Dittel, T., Aryus, H.-J.: How to “Survive” a Safety Case According to ISO 26262. In: Schoitsch, E. (ed.) SAFECOMP 2010. LNCS, vol. 6351, pp. 97–111. Springer, Heidelberg (2010)
3. Hamann, R., Sauler, J., Kriso, S., Grote, W., Mössinger, J.: Application of ISO 26262 in distributed development ISO 26262 in reality, SAE Technical Paper (2009)
4. Born, M., Favaro, J., Olaf, K.: Application of ISO DIS 26262 in practice. In: Proc. of the 1st Workshop on Critical Automotive Applications: Robustness & Safety (2010)
5. Schubotz, H.: Experience with ISO WD 26262 in Automotive Safety Projects, SAE Tech. Paper (2008)
6. Hillenbrand, M., Heinz, M., Adler, N., Müller-Glaser, K.D., Matheis, J., Reichmann, C.: ISO/DIS 26262 in the Context of Electric and Electronic Architecture Modeling. In: Giese, H. (ed.) ISARCS 2010. LNCS, vol. 6150, pp. 179–192. Springer, Heidelberg (2010)
7. Johannessen, P., Halonen, Ö., Örsmark, O.: Functional Safety Extensions to Automotive SPICE According to ISO 26262. In: O’Connor, R.V., Rout, T., McCaffery, F., Dorling, A. (eds.) SPICE 2011. CCIS, vol. 155, pp. 52–63. Springer, Heidelberg (2011)
8. Hillenbrand, M., Heinz, M., Müller-Glaser, K., Adler, N., Matheis, J., Reichman, C.: An approach for rapidly adapting the demands of ISO/DIS 26262 to electric/electronic architecture modeling. In: Proc. of the Intl. Symp. on Rapid System Prototyping (2010)
9. Makartetskiy, D., Pozza, D., Sisto, R.: An Overview of software-based support tools for ISO26262. In: Intl. Workshop Innovation Inf. Tech. - Theory and Practice (2010)
10. Hillenbrand, M., Heinz, M., Adler, N., Matheis, J., Müller-Glaser, K.: Failure mode and effect analysis based on electric and electronic architectures of vehicles to support the safety lifecycle ISO/DIS 26262. In: Intl. Symp. on Rapid System Prototyping (2010)
11. Schubotz, H.: Integrated safety planning according to ISO 26262, SAE Tech. Paper (2009)
12. Palin, B., Ward, D., Habli, I., Rivett, R.: ISO 26262 safety cases: compliance and assurance. In: IET Intl. System Safety Conf. (2011)
13. Conrad, M., Munier, P., Rauch, F.: Qualifying Software Tools According to ISO 26262. In: Model-Based Development of Embedded Systems (2010)
14. Hillebrand, J., Reichenpfader, P., Mandic, I., Siegl, H., Peer, C.: Establishing Confidence in the Usage of Software Tools in Context of ISO 26262. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 257–269. Springer, Heidelberg (2011)
15. Robinson-Mallett, C., Heers, H.: Qualifizierung der Konfiguration eines Integrations-HiL zum Nachweis einer Fahrerassistenzfunction im Kontext der ISO 26262. In: Elektronik im Kraftfahrzeug, Internationaler Kongress mit Fachausstellung (2011)

16. Wang, Q., Wallin, A., Izosimov, V., Ingelsson, U., Peng, Z.: Test tool qualification through fault simulation. In: European Test Symp. (2012)
17. Åström, A., Izosimov, V., Örsmark, O.: Efficient software tool qualification for automotive safety-critical systems. In: Elektronik im Kraftfahrzeug, Internationaler Kongress mit Fachausstellung (2011)
18. Arkley, P., Riddle, S.: Overcoming the traceability benefit problem. In: Proc. of the 13th IEEE Intl. Conf. on Requirements Engineering (2005)
19. Andersen, B.S., Romanski, G.: Verification of safety-critical software. ACM Queue 9(8), 1–10 (2011)