On the Need of a Methodological Approach for the Assessment of Software Architectures within ISO26262

Valentina Bonfiglio[1], Leonardo Montecchi[1], Francesco Rossi[2], Andrea Bondavalli[1]

[1]Dipartimento di Matematica e Informatica, University of Firenze
Viale Morgagni 65, I-50134, Firenze, Italy
{valentina.bonfiglio,lmontecchi,bondavalli}@unifi.it

[2]ResilTech s.r.l.
Piazza Iotti 25, I-56025, Pontedera, Italy
francesco.rossi@resiltech.com

Abstract. Safety analysis is becoming more and more important in a wide class of systems. In the automotive field, the recent ISO26262 foresees safety analysis to be performed at different levels: system, software and hardware. The assessment of architecture with respect to safety is typically better understood at system and HW levels, while an equivalent analysis at SW level has not such an established background. In literature, approaches exist to handle specific activities related to the safety assessment of software, but they are typically not so well integrated within a more general assessment and certification process. Recent safety standards put more and more emphasis on software-level safety analysis, therefore calling for a precise methodology for the assessment of software architectures. While ISO26262 requirements prescribe safety analysis of the software architecture, clear guidelines on how it should be performed are not provided, thus leaving an important gap for its industrial adoption. In this paper we provide our view on how such analysis should be performed, through the identification of well defined and repeatable activities, thus providing our contribution to a timely problem of great relevance in the automotive domain.

1 Introduction

Safety analysis supports the production of convincing evidence that the operation of the system is safe, i.e., even in presence of failures, catastrophic consequences on the user(s) and the environment are avoided [1]. To this purpose, the system architecture is typically analyzed using systematic techniques like Failure Modes and Effects Analysis (FMEA) [2] to identify possible violations of safety requirements.

The importance of rigorous methodologies to perform safety analysis is increasing, since the complexity of modern safety-critical systems and their dependence on electronic components are growing. As a consequence, software is becoming more and more important in the design of safety-critical systems, as more and more safety requirements are assigned to it. Indeed, safety standards are starting to put more emphasis on software-level safety analysis. Indeed, the recent standard ISO26262 [3] for the

adfa, p. 1, 2011.

functional safety of road vehicles foresees safety analysis to be performed at different levels: system, hardware, and software. In the future, a similar shift may occur in other domains as well. It is worth to specify that within the standard, the term “safety analysis” identifies a precise activity: the study of faults, their correspondent effects and the possible mitigations to be introduced.

While in performing safety analysis it is common practice to consider both hardware and software, the assessment of architecture with respect to safety is typically better understood at system and hardware levels, while an equivalent analysis at software level has not such an established background. Safety analysis of software introduces significant challenges with respect to the hardware counterpart: for example, failure modes and related statistics aren’t typically available as datasheets. Moreover, even small changes to the software architecture or to its components can produce significant effects on the propagation or mitigation of failures.

While ISO26262 requirements prescribe safety analysis of the software architecture to be performed, clear guidelines on how such analysis should be performed are not provided, thus leaving an important gap for its industrial adoption. Several companies in the automotive industry are adapting to ISO26262 requirements on safety analysis; however, public information on how they accomplish such task, and the current progress of this activity are typically not publicly available. The aim of this paper is to clarify how such analysis should be performed in order to fulfill the requirements of ISO26262, through the definition of a workflow composed of well defined and repeatable activities.

The paper is organized as follows. Related work is discussed in Chapter 2, focusing on safety analysis of software architectures, as well as previous publications on the ISO26262 standard. Chapter 3 describes our workflow for the safety analysis at software level, and how it relates with the requirements of the ISO26262 standard. Finally, concluding remarks are reported in Chapter 4.

2 Related Work

In literature, the topic of safety analysis of software architectures has been addressed in different ways. Most work focuses on methods and tools to support the application of FMEA at software level (SW-FMEA). A well-known approach is based on failure propagation and transformation annotations. The design specification of the software architecture is annotated with information about the failure behavior of the architectural components. Different notations supporting such approach exist; and one notable example is Fault Propagation and Transformation Calculus (FPTC) [9]. Using such an approach, the failure behavior of the entire system can be automatically calculated starting from the failure behavior of its components and the design of the software architecture. Further details and comparison of such approaches can be found in [8].

Some approaches focus on detailed software FMEA, i.e., they take into account a code-level representation of software components and perform a qualitative analysis of software, based on tracing the dependencies across variables through a body of source code. Other works, e.g. [6], focus on improving the manipulation of data involved in the safety analysis process.

One of the most comprehensive methods for safety analysis of system and software architectures is the Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) methodology [7]. HiP-HOPS modifies and integrates classical safety analysis techniques, guiding the analysis from the functional level through low levels of its hardware and software implementation, and provides support for the automation of certain tasks (e.g., the construction of fault-trees).

In this paper we focus on a set of requirements dictated by the ISO26262 standard to define a systematic methodology that is able to fulfill them during the assessment process. Our objective in this paper is not to introduce novel analysis methods, but rather to precisely define the set of needed activities, together with their inputs and outputs, and organize them in a structured workflow. In this perspective, our proposal is complementary to other works mentioned above, which can be used to carry out specific activities within the workflow.

A number of recent publications have targeted the ISO26262 standard, including introductions to the standard itself [11], experience reports [12], support tools [12]. Other publications focus on specific aspects of system development and assessment according to ISO26262. The work in [5] introduces a set of best practices for model review of software models with the aim of ensuring safety-related objectives and adherence to ISO26262, using a combination of automated and manual reviews. As mentioned above, the work in [4] addresses the formalization of requirements, targeting the EAST-ADL language within the ISO26262 context. A more comprehensive survey on recent publications related to ISO26262 can be found in [10].

Despite the relatively large number of publications on the new automotive standard, a proven workflow to properly support the safety evaluation of software architectures according to ISO26262 requirements is still missing from industrial practice. In this paper we provide our contribution in filling this gap.

3 Assessment of Software Architectures according to ISO26262

The system lifecycle described in the ISO26262 standard foresees safety analysis to be performed at different levels: system, SW and HW. The importance of carrying out a safety analysis at the SW level is highlighted in Part 6 of the standard, “Product development at the software level”. In particular, within the SW lifecycle, the activity of software safety analysis can be contextualized within the “Software architectural design” phase, described in Clause 7 of Part 6 [3]. In this section, the specific clauses of each part of ISO 26262 are illustrated in the reference phase model for the SW development.

According to requirement 7.4.13 (Part 6), safety analysis of the software architecture has the aim to identify or confirm the software components on which are instantiated requirements or that otherwise have an impact on them, and to support the specification of safety mechanisms and the verification of their efficiency. In this regard the standard suggests some mechanisms for error detection and error handling that should be defined at the software architectural level.

Part 6 explicitly states that safety analysis should be performed at the software architectural level (requirement 7.4.13), and references Part 9 for further guidance on this topic. Part 9 of the standard, “Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses”, however, defines the requirements and recommendations at a generic level and does not provide specific recommendations regarding software.

Summarizing, while the standard explicitly requires that safety analysis is performed on the software architecture, no details on how it can be performed from a practical point of view are provided, favoring ambiguity on activities to be performed from a practical point of view. From an industrial perspective, a well-defined and repeatable methodology is paramount, since it allows reducing efforts and costs in the assessment and certification process. The idea of the methodology, that we are developing, is summarized as a flow of activities in Fig. 1.

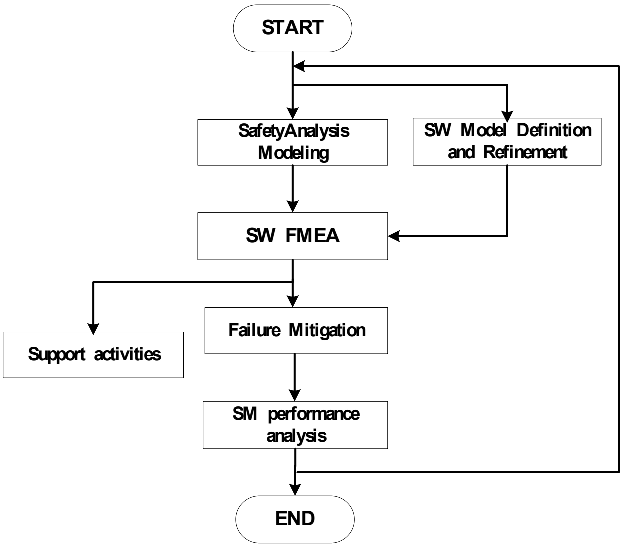

Fig. 1. Our workflow for safety analysis of software architectures according to ISO26262.

Mandatory inputs of the analysis are the *software safety requirements*, and the *software architecture information.* Accordingly, the first activity in the workflow, **Safety Analysis Modeling**, receives as input the safety requirements of software, and produces as output a set of properties that should be represented in the model. This also defines the fault models to be considered for software components and it covers requirement 8.4.6 in Part 9, which requires to define fault models consistent with the appropriate design phase, and part of requirement 8.4.9, which prescribes a systematic identification of faults and requires the evaluation of the consequences of each identified fault to determine its potential to violate safety requirements. Requirement 8.4.9 in Part 9 requires that both the software component itself and the interaction with others are considered. Depending if the software manufacturer provides some kind of model or not, the model may need to be created, or enriched; therefore an additional activity of **SW Model Definition and Refinement** is needed.

The next activity consists in performing an architectural-level **SW FMEA**. Its objective is to evaluate the impact of software faults, estimating the ability of the software to provide protection from the effects of software failures. The results of the SW FMEA describe the possible effects of software faults on the system, and indicate if safety goals or safety requirements assigned to software are complied with, as required by requirements 8.4.2 and 8.4.9 in Part 9. If SW failures are still present, a **Failure Mitigation** activity is required in order to derive mechanisms that are able to prevent, mitigate, or reduce the effect of the potential safety requirement violation. This activity meets several requirements present in Part 6 and Part 9 of the ISO26262 standard. On one hand, the specification of safety mechanisms is one of the main objectives of safety analysis, according to requirements 7.4.13, 7.4.14 and 7.4.15 in Part 9. On the other hand, requirement 8.4.3 in Part 9 explicitly states that, if a safety goal or safety requirement is not satisfied, the result of the analysis should be used to derive prevention, detection, or mitigation measures. Also, it may be necessary to determine additional safety-related test cases (requirement 8.4.7, Part 9) in order to provide evidence of correct behavior.

After the introduction of mitigation mechanisms, a support activity, **Safety Mechanism performance analysis**, is started. This provides us evidence of the performance of SMs that are currently defined and if necessary, proposes to evaluate alternative solutions as expected from requirement 7.4.13 in Part 6 and requirement 8.4.9 in Part 9. This activity is intended mainly to drive the specification of SMs, i.e., if a SM is not effective it is possible to remove it. Of course, validation of implemented SW is done at later stage. If changes are made to the architecture or safety mechanism are added, the entire workflow must be repeated from the beginning.

In order to satisfy other requirements related to safety analysis, additional support activities are identified. The **HA&RA Support** activity is related to requirements 7.4.16 in Part 6 and 8.4.5 in Part 9, and its output provides a feedback to other phases of the safety lifecycle (not shown in the figure). The **Interference Analysis** activity, which is related to requirement 7.4.13 in Part 6, consists in verifying that failures of lower integrity modules do not have impact on higher integrity modules.

4 Concluding Remarks

The importance of safety analysis of software architectures is continuously growing, since more and more functionalities of safety-critical systems are being implemented by electronic devices. In particular, the recent ISO26262 standard comprises several requirements on the safety analysis of software; however it does not provide clear guidelines on how such requirements should be fulfilled. Defining a precise workflow for the assessment of software architectures is therefore of great industrial relevance.

In this paper we emphasized the need of a structured workflow, and we have proposed a high-level view of the activities that are needed to perform a rigorous safety analysis in accordance with ISO26262 requirements. We believe that this work provides useful insights to the automotive domain, and contributes to the solution of a timely problem of great industrial relevance. We are currently working on the practical development of this workflow, aiming at its automation and integration into a tool to support the safety analysis in the automotive domain.

Acknowledgment

This work has been partially supported by the TENACE PRIN Project (n.20103P34XC) funded by the Italian Ministry of Education, University and Research, and by the MS-VIVA and RACME-MAAS projects, funded by the Tuscany Region within the framework POR CReO FESR.

References

1. Avižienis, A., Laprie, J.-C., Randel, B, Landwehr, C.: Basic Concepts and Taxonomy of Dependable and Secure Computing, IEEE Transactions on Dependable and Secure Computing, 1, 11-33 (2004).
2. Palady P.: Failure modes and effects analysis. PT Publications, West Palm Beach, FL, (ISBN: 0-94545-617-4) (1995).
3. ISO: International Standard 26262 Road vehicles -- Functional safety (2011).
4. Bergenhem, C., Johansson, R., Lönn, H: A Novel Modelling Pattern for Establishing Failure Models and Assisting Architectural Exploration in an Automotive Context. In: Computer Safety, Reliability, and Security, LNCS, vol. 7612, pp 247-257 (2012).
5. Stürmer, I., Salecker, E., Pohlheim, H.: Reviewing Software Models in Compliance with ISO 26262. In: Computer Safety, Reliability, and Security, LNCS, vol. 7612, pp 258-267, (2012).
6. Bicchierai, I., Bucci, G., Nocentini, C., Vicario, E.: An Ontological Approach to Systematization of SW-FMEA. In: Computer Safety, Reliability, and Security, LNCS, vol. 7612, pp 173-184 (2012).
7. Papadopoulos, Y., McDermid, J., Sasse, R., Heiner, G.: Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. Reliability Engineering & System Safety, vol. 71, Issue 3, Pages 229-247 (2001).
8. Grunske, L.; Han, J.: “A Comparative Study into Architecture-Based Safety Evaluation Methodologies Using AADL's Error Annex and Failure Propagation Models”. In 11th High Assurance Systems Engineering Symposium (HASE 2008). pp. 283-292 (2008).
9. Wallace, M.. “Modular architectural representation and analysis of fault propagation and transformation. Electr. Notes Theor. Comput. Sci., 141(3):53–71 (2005).
10. Izosimov, V., Ingelsson, U., Wallin, A.: Requirement decomposition and testability in development of safety-critical automotive components. In: Computer Safety, Reliability, and Security. LNCS vol. 7612, pp 74-86 (2012).
11. Dittel, T., Aryus, H.-J.: How to “survive” a safety case according to ISO 26262. In: Computer Safety, Reliability, and Security, LNCS vol. 6351, Springer, pp. 97-111 (2010).
12. Schubotz, H.: Experience with ISO WD 26262 in Automotive Safety Projects, SAE Tech Paper (2008).
13. 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).