2010 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops

An industrial case study using an MBE approach: from architecture to safety analysis

Stefano Dalpez, Roberto Passerone
Dipartimento di Ingegneria e Scienza dell'Informazione
University of Trento,
Italy,
roberto.passerone@unitn.it, stefano.dalpez@gmail.com

Daniela Cancila, Francois Terrier
CEA, LIST,
Laboratory of model driven engineering for embedded systems,
Point Courrier 94, Gif-sur-Yvette, 91191,
France
cancila@gmail.com

***Abstract*—We discuss the initial phases of software development of a real industrial safety-related device in the railway application domain. In particular, to achieve greater confidence in the system, we illustrate the development of the system architecture (using a standard model domain-specific language), the computation of the safety integrity level and the calculation of the reliability of the whole system. We reiterate the safety analysis on the sub-systems. The proposed methodology has found immediate industrial applications.**

***Keywords*-railway application domain; preliminary hazard analysis; reliability analysis; model-based safety engineering**

I. Introduction

The development of critical systems involves the interplay of many different disciplines and, therefore, becomes particularly complex. In this context, the industrial and academic communities are paying increasing attention to safety issues. Because safety concerns may involve profound changes in the architecture of a system, the values of the safety attributes must be calculated as soon as possible during development. This is the case, for example, for the *Safety Integrity Level* (SIL) [1], wich is related to the degree of failures that the system must be able to tolerate.

The industrial development of a critical system involves its successive refinement until the program code. A refinement produces logical sub-systems that are described by their related documentation. Then, safety analysis is performed iteratively on the subsystems, by following the norms related to the considered application domain. The result is in the form of documentation which includes the values of the safety attributes. Finally, the whole process is certified by a third authority.

Even excluding certification, this process presents many difficulties. For example, it must guarantee that the produced documentation on the architecture and the safety analysis is coherent, and that the safety analysis documentation is correct w.r.t. the architecture. In addition, safety norms clearly point out the adoption of semi-formal languages as a *means* to improve the safety analysis.

The main objective of this work, then, is a feasibility study. We pay a special effort to be adherent to the standard languages for the architectural (sub)system specification and for the corresponding safety analyses. More in particular, we discuss the development of the so-called *Event Recorder*, a real industrial case study in the railway application domain. Its main functionality is to periodically memorize the state of the system, so that, in case of an accident, the potential causes could be extracted and analyzed. We focus on the initial phases of architectural design of the Event Recorder system, we perform the related safety analyses by computing the Preliminary Hazard Analysis and the reliability analysis of the whole system (that is, the safety analyses include the architectural subsystems). Our adopted methodology adheres to the current norms [1], [2], and reduces the cost linked to the industrial application, because the safety analysis is performed before the system is developed in all its complexity. The main conclusion of our study is that existing specification languages already provide constructs to specify the safety attributes during the architecture refinements of a system - thus integrating the safety analysis results. However, the main drawback is the lack of safety analysis tools, which are directly integrated on an architectural model specification.

II. Methodology

Table I shows our methodology. Each line shows a level of the developed architectural system, the corresponding safety analysis, the main safety attributes, which must be calculated by the safety analysis, and the corresponding constraints given by safety objectives.

In Step (a1), designers specify the system architecture by using a model-based engineering (MBE) approach [3], [4]. MBE provides the mechanisms to abstract away unnecessary details and, hence, to facilitate the design phase, the validation and verification processes, reuse and evolution. In Step (a2), safety designers annotate the architectural model with safety attributes, according to the Preliminary Hazard Analysis (PHA) [2]. PHA is performed by traditional techniques, but the results are annotated in the safety attributes, which are already introduced in the architecture - thus integrating PHA with the architecture. As a result, designers construct

978-0-7695-4038-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ISORCW.2010.11

Architecture	Safety Analysis	Achieved Safety Values	Constraints given by Safety Objectives
(a1) (Safety-Related) Event-Recorder System	(a2) Preliminary Hazard Analysis	(a3) SIL	(a4) The architecture should be reduntant to at least one fault
(b1) (Safety-Related) Event-Recorder System with Architecture 1oo2	(b2) Reliability Analysis	(b3) PFH, MTBF	(b4) the PFH and MTBF values should correspond to the SIL value; the architectures (e.g. (b1) and (c1)) should always meet the MTBF value
(c1) (Safety-Related) Event-Recorder Subsystems: Power subsystem and Event-Recorder Functional Subsystems	(c2) Reliability Analysis	(c3) MTBF for each analysed subsystem (e.g. Power and Event-Recorder Functional subsystems)	(c4) the pair of the MTBF values should meet the MTBF value of Event-Recorder system

Table I
OVERAL METHODOLOGY

a complex model where the architecture is *coupled* with the safety model and analysis.

The Safety Integrity Level (SIL) value for the safety-related functionalities of the system is computed during PHA (Step (a3)). SIL is a standard attribute required by the IEC61508 standard [1]. At this point, the system designers must refine the architecture to satisfy the SIL value (Step (b1)). Such refinement is required by the IEC61508 standard [1] (Step (a4)). Then, reliability analysis calculates the probability that a device will perform its required function under stated conditions for a specific period of time (Step (b2)) [5], and provides the value of *Probability of Failure per Hour* (PFH) [1] and the corresponding value of *Mean Time Between Failure* (MTBF) [1] (Step (b3)).

The system is in a safe state if the following two conditions are guaranteed: the PFH and the corresponding computed MTBF value belong to a real-number interval derived from the SIL value (Step (b4)) [1]; and the architecture, together with its refinements, guarantees that the MTBF value is satisfied (Step (b4)) [1]. To ensure this, the MTBF value is decomposed (Step (b3)) and allocated (Step (c1)) to the subsystems, and each subsystem must be shown to satisfy the allocated MTBF (Step (c4)). The MBTF decomposition is a semi-qualitative analysis based on the experience of the safety designers. A graphical representation of the reliability analysis for the subsystems is made difficult by the multi-dimensional nature of the space, thus complicating the visualization of the result. However, our methodology ensures consistency between the different levels of abstraction during the system refinements (Steps (a1), (b1) and (c1)). The safety values, which are calculated by the safety analyses (Steps (a3), (b3) and (c3)), are annotated in the architecture (Steps (b1) and (c1)) and, hence, they can be proved to be preserved (Steps (c4), (b4), (a4)). In the next sections, we first discuss the related work and, then, we detail each step by means of a case study, based on a railway industrial application.

III. RELATED WORK

Of the extensive literature on safety-related issues, we here discuss existing work that is closer to our approach and which uses safety-related languages. The first three works in this section adopt both UML and UML profiles, to specify the architectural system, and they introduce a safety-related domain-specific language as UML profile, to deal with safety analysis.

De Miguel et al. propose a UML standard safety profile that integrates safety analysis and a UML architecture [6]. The profile allows safety engineers to annotate a UML architecture with safety attributes and, thanks to model transformations, Fault-Tree Analysis (FTA) and Failure Mode Effects and Criticality Analysis (FMECA) are automatically generated. Currently, the work of de Miguel et al. does not consider the recent trend to specify the architecture with the two UML standard profiles: SysML and MARTE. Such a trend is captured by SOPHIA [7].

SOPHIA provides a modeling language to integrate safety in a model-based approach [7]. The work introduces a metamodel and a profile by intentionally reusing the pre-existing MARTE stereotypes - thus ensuring compositionality between the language, which is used to specify the architecture, and the language, which is used to specify safety models and safety analysis. Moreover, SOPHIA allows safety designers to achieve a model representation of safety issues, for example a model of the accidents to be avoided. The safety model couples with the architectural model - thus guaranteeing the coherence between the architectural system and safety analysis of that architectural model. Currently, SOPHIA does not calculate the SIL and the tests are only given on a flat (real industrial) system (that is, without considering the refinement of a system in its subsystem). Unfortunately, we were not authorized to use SOPHIA on (a subset of) our example, due to intellectual property limitations.

DAM profile introduces a UML profile for quantitative dependability analysis of architectural systems [8]. DAM extends MARTE and contains three main models: the System Core model, the Thread model and the Maintenance model. In the DAM Thread model, the information on error

propagation from a faulty component to another component is transmitted via the corresponding connector. Then, safety-related annotations are specified on components and on connectors. Some safety-related attributes, which we use, are included in the Thread model, but the DAM model is not complete for our needs. Moreover, a tool that implements the safety analysis starting from the model specification is not currently available.

Our approach is closer to the mentioned work because it adopts UML2 and UML profiles. Despite some differences, which we will discuss in the sequel, we point out that all these approaches are potentially compatible and might therefore be used in combination.

The primary objective of our methodology is to perform safety analysis of an architectural system and propagate the results back in the architectural model. Indeed, safety analysis results are directly introduced in their corresponding safety attributes, which are specified in the architecture. This ensures that safety analysis results are available in the architectural system, by preserving the safety values during the refinement of the system in its subsystems. We are not interested in introducing a new safety-related domain-specific language, but we wish to reuse as much as possible pre-existing UML standard profiles. More in particular, we adopt the following specification languages: UML2, SysML and MARTE (the UML2 standard profile for Modeling and Analysis of Real-Time Embedded systems, packages VSL and NFP) to specify the architectural model, and MARTE::VSL to specify safety attributes and their values on the architectural system components.

We conclude this section by discussing three standards: Architecture Analysis & Design Language (AADL), AUTOSAR and EAST-ADL .

Error propagation is captured by the standard Architecture Analysis & Design Language (AADL) [9]. AADL is a formalism which is particularly suitable for high-integrity embedded systems, like spatial applications. AADL provides relevant features on deployment and error propagations. Other relevant standards are AUTOSAR (the European industrial standard to specify component-based software infrastructures in automotive applications [10]) and EAST-ADL [11], which is used by the scientific community to define an architecture description language and to complement AUTOSAR. EAST-ADL focuses on automotive applications. The main advantages of these standards are the adherence to the automotive norms, and their methodology to software development. We have not addressed the AADL, AUTOSAR and EAST-ADL standards because we do not deal with the same application domain. Indeed, our main effort is in the safety analysis of a real industrial railway system such that the safety analysis is compliant to the railways norms, e.g., the IEC61508 standard [1], and the architectural system is based on a MBE approach and it is specified in UML and UML profiles.

IV. Case Study

We study an Event Recorder system whose main functionality is to periodically memorize the state of the system. The recorded information plays a crucial role in case of an accident, because it allows engineers to recover the system state immediately before the accident happened, and to reconstruct the causes of the accident. According to the requirements, which are given us by the industry, we deploy in the Event Recorder system two safety-related functionalities: the deadMan functionality, which ensures the safety of passengers in case the machinist should become unable to operate the commands, due for instance to a sudden illness; and the zeroVelocity functionality, which ensures that the doors are locked whenever the train speed is slightly higher than zero.

We call *Safety-Related Event Recorder* the Event Recorder system that implements the two mentioned functionalities.

The Safety-Related Event Recorder has five main components. The **Zero Velocity (ZV) component** guarantees that the doors are locked whenever the train speed is slightly higher than zero. ZV records the velocity sent by two sensors placed at the opposite ends of the train. ZV checks that the two measures have the same value. If the values are different, it sends an error message to the Emergency Break component, which must stop the train. Otherwise, if the velocity is zero, it inhibits the DeadMan component. Finally, it periodically updates its state in the Buffer component.

The **Dead Man (DM)** component guarantees the safety of passengers in the case the machinist becomes unable to operate the commands, due for instance to a sudden illness. The DM component provides three services: machinistController, inhibitor and errorDetection. The machinistController service receives a signal from a sensor, which interfaces with the machinist. If the signal is not periodically received, then the DM component sends an error message to the Emergency Break component, which must immediately stop the train. The inhibitor signal, on the other hand, inhibits the machinistController service; that is, when the train is stopped, for example at a platform, the machinist does not have to give the vital signal to the sensor. The errorDetection signal is generated by the Diagnostic component and is used to send an error message to Emergency Break component, which must stop the train. Finally, the DM component periodically updates its state in the buffer component. The **Diagnostic component** periodically executes three actions. First it reads the states of the ZV component and DM component in the buffer. Then, it verifies the critical functionalities of the system and, finally, it sends the information to the external components via a bus. If the Diagnostic component detects an error, it sends an error signal to the DM component via the errorDetection service.

The **Buffer component** simply records the system state, and is written by the DM and ZV components, and read by

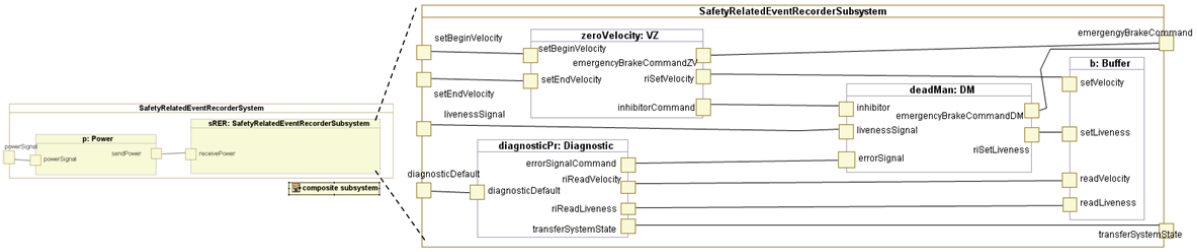

Figure 1. Composite Diagram for Safety-Related Event Recorder System

the Diagnostic component.

Finally, the **Power component** supplies power to the Safety-Related Event Recorder system. In order to meet safety requirements, the Power component must receive the power from external components instead of the backplane, to the safety-related Event Recorder system and, then, transfer the received power to the internal components.

A. Architecture - Steps (a1), (b1), (c1)

MBE is supported by a large variety of model-based languages [12], [13], [14]. In order to specify a systems, we must account for the interplay of several features, including architecture, safety, and temporal attributes. Moreover, our modeling language should be supported by tools that allow us to analyze if the system meets the systems requirements.

Several methodologies and approaches to introduce a modeling language can be found in the literature. Some languages are introduced from scratch for specific needs. Their main advantage is to be optimally studied to the problem at hand. The price to pay is a higher cost for interfacing (meta)models and, especially, tools. This cost potentially increases in the verification and validation phase, to avoid the syntax and semantics overlapping between modeling languages [15]. Other modeling languages introduce only one metamodel, which must be as "general" as possible. Domain-specific languages are then formed as *profiles*, which extend and specialize the given metamodel. The main advantage is to reuse pre-existing plug-in and tools and, moreover, to guarantee compatibility when new plug-in and tools are integrated, leading to a lower cost of the verification and validation phase. The main drawback is the requirement to have a good knowledge of the metamodel to be extended and of the pre-existing profiles to appropriately reuse existing plug-ins [15]. In order to develop the Safety-Related Event Recorder architecture, we choose this latter approach. More in particular, we use UML2 to specify the architecture and MARTE to specify temporal attributes and non-functional properties [16], [13]. Since its introduction, MARTE has been successfully applied in several research industrial projects, especially for specifying non-functional properties. Our underlying strategy is to exploit MARTE as much as possible, in order to evaluate its expressiveness and its limitation in the specification of safety-related issues. Our results show that MARTE provides a set of constructs that are sufficient to specify safety-related values on architectural components during the first stages of the architectural development.

Figure 1 shows the UML2 composite diagram for the Safety-Related Event Recorder System. On the left, Figure 1 highlights the two components of the Safety-Related Event Recorder System (i.e., Power and the Safety-Related Event Recorder Subsystem), their relationship via the connector bewteen powerSignal port and receiveSignal port. The power component receives the power from external components via port powerSignal and, then, it transfers the power to the Safety-Related Event Recorder Subsystem via the connector between powerSignal and receiveSignal ports. On the right, Figure 1 highlights the relationship between the components within the Safety-Related Event Recorder Subsystem. The four components (also called properties in UML2 terminology) correspond to ZV component, diagnostic component, DM component and Buffer component. Consider the DM component. The DM component provides three functionalities: machinistController, inhibitor and errorDetection. The machinistController functionality is specified by a method in a provided interface, which is realized by livenessSignal port. Such provided interface is periodically invoked by a sensor, which interfaces with the machinist by sending a livness signal. If the signal is not periodically received, then the DM component sends an error message to the Emergency Break component via a delegation between the emergencyBrakeCommandDM port and the emergencyBrakeCommand port. The inhibitor signal inhibits the machinistController functionality and it is realized by inhibitor port. Like the machinistController functionality, The errorDetection signal is specified by a method in a provided interface, which is realized by errorSignal port. The method is invoked by the Diagnostic component to send an error message to Emergency Break component. Finally, the DM component periodically updates its state in the buffer component via setLiveness ports and the corresponding connector. More in general, in Figure 1, a port in the left-hand side of each component realizes a provided interface; a port in the right-hand side of each component realizes a required interface. Thereby, from a methodological point of view,

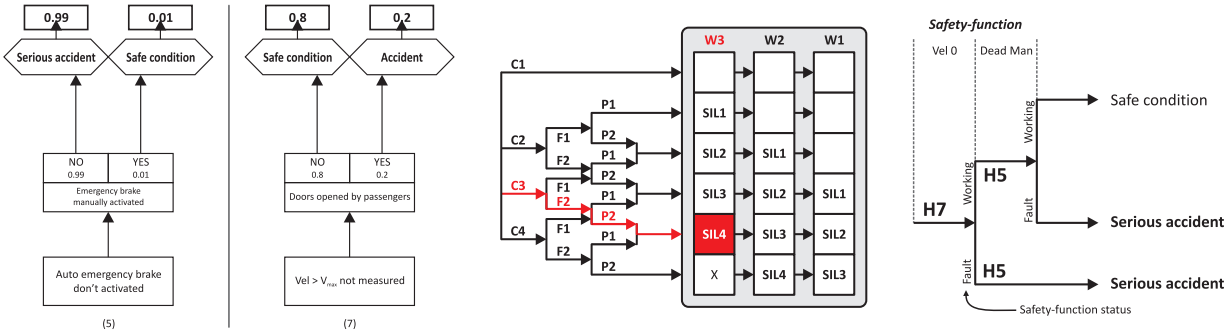

Figure 2. The consequence diagram, the SIL analysis, the functional interdependences analysis

we strictly forbid that a single port realizes both provided and required interfaces. It straightforwardly follows because our methodology is based on to the "Interface Automata" Theory [17], [18].

B. *Preliminary Hazard Analysis* - Steps (a2) and (a3)

The *Preliminary Hazard Analysis* (PHA) of a system is a preliminary analysis used to determine the Safety Integrity Level (SIL) value of each safety function of the system. The SIL is a semi-qualitative value that indicates the severity of the safety function, and is expressed as an integer number between 0 and 4 (where 4 is associated with the highest severity). Once the SIL is calculated for each safety function of the system, the safety engineers calculate the SIL value that the whole (sub)system must meet during all design refinements. For this reason, the new SIL value is often and improperly called "the SIL of the system". It is derived by studying the interdependences between the mechanisms (or barriers) that are introduced in the system to mitigate the severity of potential accidents.

The Safety-Related Event Recorder system has only two main safety-related functions: the machinistController, which is realized by the DeadMan component, and setVelocity, which is calculated by the ZeroVelocity component by comparing the setBeginVelocity and the setEndVelocity values.

In the sequel, we briefly discuss the semi-qualitative analysis to calculate the SIL. First of all, we study the probability that when a given accident has just happened, the mechanisms that the systems should provide to mitigate the accident are effectively used. On the left, Figure 2 shows the consequence diagram for two critical events that could happen when the train is moving: the sudden illness of the engine driver (noted by (5) in the Figure) and the incorrect measure of the velocity (noted by (7)). Consider the consequence diagram (5). The underlying context is the following: the train is moving and no other machinist is present (that is, only one engine driver is in charge of the train and one live sensor takes the machinist's liveness signals). Such a context is realistic in many cases. The safety engineers then study the probability that a passenger or other personnel pull the emergency brake as soon as the engine driver has an illness - thus reaching a safe state for the system. The probability is calculated to be less than 1%.

On the middle, Figure 2 shows how the SIL is calculated for the machinistController functionality, which is realized by the DeadMan component. The adopted methodology is based on the IEC61508 and EN50126 standards [1], [2] and can be only used in "simple or self-evident" systems [2]. The Safety-Related Event Recorder system is a typical example in which we can apply such a methodology. In the figure, C specifies the severity of the accident associated to the case in which an engine driver has an illness; F specifies the impact of the accident on passengers; P is the probability that a passenger uses a given mechanism to reduce the severity of the accident; finally W specifies if the system provides mechanisms (barriers) that should be used by passengers to mitigate the accident. The value of the mentioned parameters is set with respect to the EN50126 standard [2]. We apply the same methodology to calculate the SIL of the second safety-related function. And then we set the SIL value to 2.

In order to calculate the SIL value that the whole (sub)system must meet during all design refinements (Steps (a1), (b1) and (c1)), the safety engineers analyze the interdependencies between the safety-related functions [1] and the mechanisms (or barriers) that are introduced in the system to mitigate the severity of potential accidents. On the right, Figure 2 shows such an analysis for the Safety-Related Event Recorder system. The system could involve a potential accident if one of the following scenarios happens: the setVelocity functionality and the machinistController have a failure (line: H7 fault +H5 fault); the setVelocity functionality works but the machinistController has a failure (line: H7 working +H5 fault). The safety-related functionalities of the Safety-Related Event Recorder system have different SIL values (SIL 4 and SIL2). But the analysis of the interdependencies shows that the system is in a safe state only if the "SIL of the system" is set to 4.

1) Architectural Constraints given by Safety Objectives - Steps (a4) and (b1) *:* SIL4-functionalities must be realized by

a redundant architecture to guarantee fault tolerance to at least one failure - according to the IEC61508 standard [1] (Step (a4)).

Architecture 1oo2 provides an example of a redundant architecture, which is compliant to the IEC61508 standard [1]. The architecture owns two subsystems, each able to realize the safety-related functionalities. Since the Safety-Related Event Recorder System is made of one single hardware unit, we need two such units (Step (b1)).

We adopt Architecture 1oo2 for three main reasons. First of all, it is easy to realize. Secondly, we skip the well-known problems due to the critical circuits when three or more units are deployed on the same system. Finally, Architecture 1oo2 is an economically viable solution because it consists of only two identical subsystems.

Of course, in Architecture 1oo2 we should pay attention to the common failures that crash the whole system. We discuss such an analysis in the next section.

C. *Reliability Analysis* - Steps (b2)-(b4)

Reliability is defined as the probability that a device will perform its required function under stated conditions for a specific period of time [5]. Reliability is quantified as Mean Time Between Failures (MTBF) and is based on the combination of three main parameters: detected faults; undetected faults and common faults.

Detected faults are detected by the diagnostic components, which are integrated, in our case, in the Architecture 1oo2. *Undetected faults* are not detected by the diagnostic components - thus they increase the hazard of a potential accident. Then, the probability of failure per hour of the undetected faults should be guaranteed to be under a threshold, specified by the norms. Finally, the *common faults* potentially crash the whole system.

Equation (1) defines the *Probability of Failure per Hour* (PFH) in accord to the IEC61508 standard [1] . The meaning of the acronyms of the equation is extracted by the IEC61508 standard [1] and introduced in Table 1:

$$PFH = 2((1-\beta_D)\lambda_{DD}+(1-\beta)\lambda_{DU})^2 t_{CE}+\beta_D\lambda_{DD}+\beta\lambda_{DU} \quad (1)$$

The last column of Table 1shows the safety values, which the architecture must meet. They represent our industrial safety requirements. In the table, the first four values are determined on the basis of the safety engineer experience. They are qualitative values that range in a fixed interval, given by the IEC61508 standard [1]. The last four values depend on the value of the MTBF, which is equal to $\frac{1}{\lambda}$ in most cases.

Figure 3 shows the MTBF value that the Safety-Related Event Recorder system must meet to be safe. The MTBF value is obtained for the values in Table 1 (last column), and it is given from the intersection of the curve line with the threshold between the SIL3 and SIL4 [1]; that is, 1,100,000. The MTBF value and the values in Table 1 are used in Equation 1 - thus calculating the PFH value. The IEC61508 standard [1] introduces suitable mappings between PFH and SIL: The system satisfies the quantitative requirements of SIL level if the MTBF value belongs to the PFH real interval corresponding to SIL 4 [1].

Param.	**Meaning**	Value
PFA	Probability of Failure par Hour	
β	The fraction of undetected failures that have a common cause	5%
β_D	The fraction of those failures that are detected by the diagnostic tests, the fraction that have a common cause	2%
DC	Diagnostic coverage	95%
$MTTR$	Mean time to restoration (hour)	1 h
$MTBF$	Mean Time Between Failure	
λ	Failure rate (per hour) of a channel in a subsystem	$\frac{1}{MTBF}$
λ_{DD}	Detected dangerous failure rate (per hour) of a channel in a subsystem	$\frac{\lambda}{2}DC$
λ_{DU}	Undetected dangerous failure rate (per hour) of a channel in a subsystem	$\frac{\lambda}{2}(1-DC)$
λ_D	Dangerous failure rate (per hour) of a channel in a subsystem,	λ_{DU} + λ_{DD}

Table II
THE IEC61508 STANDARD PARAMETERS [1] AND THEIR INDUSTRIAL VALUES

1) Subsystems - Steps (c2)-(c4)*:* The MTBF value is specified in the corresponding attribute of the Safety-Related Event Recorder architectural system. The system must guarantee that the architecture will meet the MTBF value during across its refinements [1]. In order to do so, the MTBF is decomposed and allocated to the subsystems. Figure 3 shows the value of the MTBF for the whole system, as a function of the MTBF of the subsystems (the Power subsystem and the Safety-Related Event Recorder Subsystem, see Figure 1). The shaded area corresponds to the area of MTBF values for which the overall safety constraint is satisfied. We point out that in order to satisfy the overall system safety requirements, the MTBF values for the two subsystems are necessarily correlated. For example, if the MTBF for the Power component is set to 1.5e+006, then the MTBF for the Safety-Related Event Recorder subsystem must be a value between 1e+007 and 4e+006. The pair of the MBTF values represents the MTBF decomposition and it is set by the safety designers (semi-qualitative analysis). Once the MTBF values are set, these values are specified and annotated in the architectural subsystems (Figure 1).

V. DISCUSSION AND CONCLUSION

In this paper, we have discussed a real industrial example in the railway application domain. We have focused on the early stages of the software development. To specify the system architecture, we have adopted a MBE approach; the safety analysis has been performed on the system and its subsystems, and conforms to the standards EN50126 [2] and IEC61508 [1]. The main conclusion of our study is that

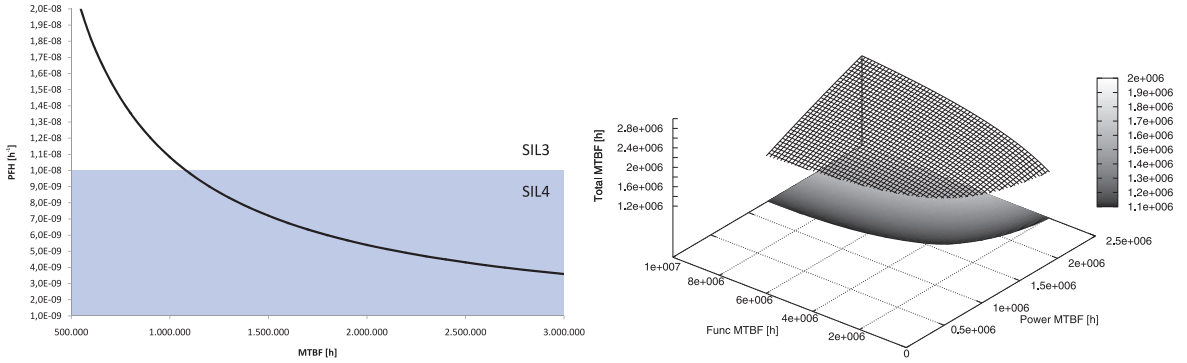

Figure 3. On the left, the MTBF value for the Safety-Related Event Record System. On the right, relationship between the MTBF value for the system and the MTBF values for its subsystems

UML and its profiles already provide constructs to specify the safety attributes during the architecture refinements - thus integrating the safety analysis results. However, the main drawback is the lack of safety analysis tools from the architectural model. Indeed, for the safety analysis, we have always exploited other tools to ensure conformance with the mentioned standards [2], [1]. Moreover, some safety attributes, such as the MTBF values, are correlated in the subsystems: if one MTBF value changes, e.g., in the Power subsystem, it can involve a change in the MTBF value of the other subsystem (and, hence, on the architecture). To capture such features, instead of modifying the UML language (and its profiles), our future work includes the development of a safety analysis tool (e.g., a plug-in in Eclipse) such that the two values are guaranteed to match.

REFERENCES

[1] IEC, *61508:1998 and 2000, part 1 to 7. Functional Safety of Electrical, Electronic and Programmable Electronic Systems.*, 2000.

[2] CENELEC, *EN-50126: Application ferroviaires - Spécification et démonstration de Fiabilité, Disponibilité, Maintenabilité et Sécurité (FMDS)*, 1999.

[3] D. Schmidt, "Model-driven engineering," *IEEE Computer*, pp. 25–31, February 2006.

[4] B. Selic, "From Model-Driven Development to Model-Driven Engineering," http://feanor.sssup.it/ecrts07/keynotes/k1-selic.pdf.

[5] S. Speaks, "Reliability and MTBF Overview," cdn.vicorpower.com/documents/quality/Rel_MTBF.pdf, Vicor Reliability Engineering, Tech. Rep.

[6] M. de Miguel, J. Briones, J. Silva, and A. Alonso, "Integration of satety analysis in model-driven software development," 2008.

[7] D. Cancila, F. Terrier, F. Belmonte, H. Dubois, H. Espinoza, S. Gérard, and A. Cuccuru, "SOPHIA: a Modeling Language for Model-Based Safety Engineering," in *Inter. Work. ACES-MB*, 2009.

[8] S. Bernardi, J. Merseguer, , and D. Petriu, "Adding Dependability Analysis Capabilities to the MARTE Profile," in *Inter. Conf. MODELS*, 2008.

[9] P. Feiler and A. Rugina, "Dependability Modeling with the Architecture Analysis & Design Language (AADL)," Software Engineering Institute, Carnegie Mellon, Tech. Rep., 2007.

[10] AUT@SAR, "Automotive Open System Architecture," www.autosar.org.

[11] ATESST Project, "Advancing Traffic Efficiency and Safety through Software Technology. ATESST STREP - FP6 project," http://www.atesst.org.

[12] SAE, "Architecture Analysis and Design Language (AADL)," www.aadl.info/aadl/currentsite/.

[13] OMG, "Unified Modeling Language UML Resource Page," www.uml.org.

[14] S. Bliudze and J. Sifakis, "The Algebra of Connectors - Structuring Interaction in BIP," in *Int. Conf. EMSOFT*, 2007, pp. 11–20.

[15] H. Espinoza, B. Selic, D. Cancila, and S. Gérard, "Challenges in Combining SysML and MARTE for Model-Based Design of Embedded Systems," in *Int. Conf. ECMDA*, vol. 5562. LNCS, 2009.

[16] OMG, "UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, v1," www.omgmarte.org.

[17] L. de Alfaro and T. A. Henzinger, "Interface automata," in *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*. ACM Press, 2001, pp. 109–120.

[18] D. Cancila, R. Passerone, T. Vardanega, and M. Panunzio, "Toward Correctness in the Specification and Handling of Non-Functional Attributes of High-Integrity Real-Time Embedded Systems," May 2010.