Data-Centric Middleware support for ASIL assessment and decomposition in open automotive systems

Jelena Frtunikj, M.Sc., fortiss GmbH, München, Deutschland
Michael Armbruster, Dr., Siemens AG, Corporate Research and Technologies, München, Deutschland
Alois Knoll, Prof. Dr. –Ing. Habil., Technische Universität München, München, Deutschland

Abstract

Current automotive electrics and electronics (E/E) architectures are very complex and face the challenge to provide easy integration of additional and even more complex functionality such as automated driving. Furthermore, the end-customer is used to the possibility of easy personalization, adaptivity or extensibility of the electronic systems with new hardware or software. Since this possibility already exists in the infotainment domain, it significantly drives a similar expectation also within the automotive domain. Due to the fact that cars are safety-critical systems, automotive middlewares (MW) must provide framework and methods for safe integration of new hardware or software. This implies whenever a new functionality or component is integrated in the system, violation of different qualitative and quantitative safety requirements needs to be evaluated.
This paper presents our recent work on runtime qualitative safety-assessment that considers Automotive Safety Integrity Level (ASIL) compatibility of signals and the possibility of their decomposition in such open automotive systems. We introduce our approach and present a case study where the approach is applied not only during the design of the automotive system but also in Plug&Play scenario, so during refurbishment or via the end-customer.

1 Introduction

The automotive industry faces the challenge to manage the E/E-architecture's complexity [1] while in parallel more and more functionality (we will call each considered SW-implemented function "service" herein) within a vehicle is required especially considering the raising demand for vehicle automation up to fully automated driving. Furthermore the end-customer is used to the possibility of easy personalization, adaptivity, extensibility of electronic systems. The infotainment-domain has already demonstrated these capabilities [11], [12] and it significantly drives a similar expectation now within the automotive domain. Thereby, the end-customer does not care about qualities such as functional safety or security, since those are expected to be ensured by default.
A possible technical approach to serve the before mentioned challenges is a so called "open automotive architecture". Core-element of this architecture is a middleware which ensures different non-functional requirements/qualities of interfaces, services and overall automotive functions. This middleware follows the design-paradigm to implement technical means to ensure quality of service properties, especially functional safety features, in a generic, function-unspecific manner. The technical measures to ensure functional safety can easily be reused for different services. This enables a quite simple development-environment for automotive functions. One just has to focus on the description of the interfaces of each service instead of developing technical measures to ensure those. However, process-measures to ensure functional safety are instead not considered to be implemented by the middleware in a generic way. Anyhow, later on within this paper a concept that shows how to verify even process requirements within a middleware during runtime is presented. This capability is essential to realize adaptivity and extensibility even in the context of safety-critical functionality.
The challenge considered within this paper is the modularization of safety-assessment (safety-verification) process. To do so, we consider the overall E/E-world of a vehicle as a set of safety elements out of context (SEooC) according to ISO26262 [3, part 10]. Then we identify the information that will be required to the safety-assessment during runtime and bundle each SEooC with this. Within the "Plug"-phase of new sensors/actors or SW-components (see later on chapter 3, figure 2), a Safety Manager component, which is a part of the middleware, preforms the safety-assessment based on the extended SEooC-description.
In this paper we consider that the overall safety-assessment can be split up to an offline and an online-part. Regarding the online-part, we differentiate requirements concerning the process-levels to which a SEooC has been developed (ASIL), timing requirements (e.g. fault-tolerance times, mission times, etc.) and quantitative probabilistic requirements (PFH, SPFM, LFM). Within this paper we mainly focus on the online-verification of the compatibility of ASILs between data-sources and sinks. Moreover, we also consider the possibility of ASIL decomposition. The remainder of this article is structured as follows. In Section 2, we first revisit some basics on the safety automotive standard ISO 26262 and the safety development process. Afterwards, we give a detailed description of the proposed concept. Section 4 presents a case study where this approach is applied. Section 5

compares our approach to available solutions provided by industry and scientific community and the last section provides a conclusion and summarizes future steps.

2 ISO 26262 overview

ISO 26262 was published in November 2011 as a functional safety standard for electrical and electronic systems in road vehicles. When applying hazard analysis and risk assessment according to ISO 26262, a set of safety goals (high level requirements) are specified and an Automotive Safety Integrity Level (ASIL) is assigned to each of them. ASIL represents the degree of rigor that should be applied in development, implementation and verification of the requirement in order to avoid unreasonable residual risk. From the top-level safety goals, more refined safety requirements are specified, which are then assigned to single hardware (HW) or software (SW) subsystems (components) of the system [3, part 3-7]. If in the design of the system architecture, sufficiently independent and heterogeneous redundant elements do exist, it is possible to allocate a specific safety requirement to two of the components. The redundant components inherit lower ASIL value than the parent. For example, an ASIL D requirement may be allocated to two independent architectural elements and the resulting two decomposed requirements inherit a lower ASIL level (e.g. ASIL B). The ISO26262 defines several rules to derive an ASIL to several independent architectural elements within part 9 and does call this “decomposition”.

Safety requirements are usually addressed by concrete countermeasures, such as fault prevention techniques (e.g. coding guidelines, development process guidelines), fault removal (e.g., testing, verification) or fault tolerance (e.g., redundancy) [4]. These countermeasures are then included in the safety concept which defines how the sound composition of different measures contributes to the achievement of the top-level safety goals. A safety case finally defines a sound argument which “proves” that the set of safety requirements is sufficiently complete and that the system complies with these requirements. The task of a certifier is to manually check all documents that are provided and come to the conclusion if the system is safe or not. In case of positive answer, the system receives the required certificate. Unfortunately, this traditional safety approach cannot be employed in the context of open automotive systems, because components/systems that are unknown at design time have to be dynamically integrated with other third-party components that already exist in the system. Hence a complete safety-assessment is impossible at development time and part of it has to be shifted at runtime and done automatically.

By the introduction of the Safety Element out of Context (SEooC) concept in the ISO26262 [3, part 10] the first steps towards enabling runtime safety-assessment (at least part of it) have been made. The ASIL capability of a SEooC defines the requirements of ISO 26262 that are applied to the development process of this SEooC and states the capability of the SEooC to comply with assumed safety requirements assigned with a given ASIL.

Therefore, the SEooC concept enables specifying safety requirements explicit at the interfaces of (sub-) systems. By using this information, safety-assessment regarding different safety attributes such as ASIL compatibility and decomposition of signals, probabilities of failures per hour (PFH) or even the hardware metrics (SPFM, LFM) can be performed.

3 Proposed Approach

This section presents the proposed concept that tackles the above stated challenges. As stated in the introduction we aim, part of the safety-assessment that checks safety requirements (regarding ASIL compatibility and decomposition) coming from different subsystems (HW or SW) and which realize a certain system service (e.g. steering functionality), to be shifted at runtime and be performed automatically. This procedure is to be executed not only when the first system configuration is generated (at design time) but also whenever new subsystems are dynamically integrated (via Plug&Play) in the open automotive system.

Before we give a detailed description of the concept, we first elaborate on the prerequisites that must be fulfilled in order to operationalize the presented approach in the context of future open automotive systems. At the end of this section we also provide some more information about the implementation details.

Prerequisites for application of the concept

A preliminary requirement for the application of the proposed method is an E/E architecture (developed according to ISO 26262 as SEooC with a certain ASIL capability) that provides Plug&Play ability. To provide Plug&Play in the automotive field even for safety critical applications, the system architecture needs to provide capabilities to guarantee extra functional properties for resource utilization, safety and security. Such architecture is suggested in the Robust and Reliant Automotive Computing Environment for Future eCars (RACE, www.projekt-race.de) project [5]. The centralized platform (Figure 1) consists of a redundant and reliable communication infrastructure based on a switched Ethernet topology, and a runtime environment (RTE). A middleware, which defines the Runtime Environment (RTE) together with the operating system, drivers, and components provides services, interconnects all system components. The MW facilitates generic safety mechanisms such as real-time deterministic scheduling, data exchange services, health monitoring and diagnosis, as well as an execution platform with time and space partitioning for applications running on the same HW and having different ASIL classifications.

In order to support Plug&Play, here the traditional message-oriented approach is replaced by a data-centric one: instead of specifying sender-receiver relationships, the subsystems developers have to specify the component interfaces by a standardized data model. Based on this data model, the MW establishes data paths between subsystems.

The data-model can be seen as a data-base full of data-elements (sensor- and actor-data) describing the vehicle's, the environment's or the driver's state.

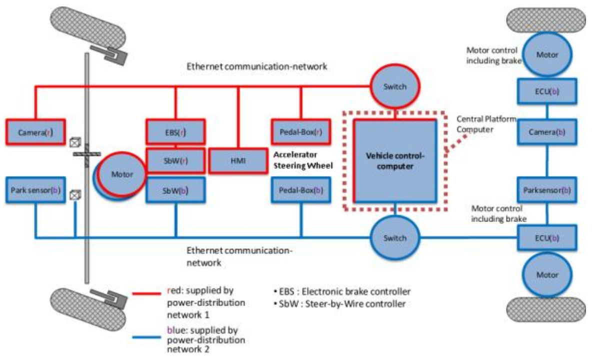
Ethernet communication-network
Switch
Motor control including brake
Motor
ECU(b)
Camera(r)
EBS(r)
SbW(r)
HMI
Pedal-Box(r)
Accelerator Steering Wheel
Vehicle control-computer
Central Platform Computer
Camera(b)
Motor
Park sensor(b)
SbW(b)
Pedal-Box(b)
Parksensor(b)
Motor control including brake
Switch
ECU(b)
Ethernet communication-network
Motor
red: supplied by power-distribution network 1
blue: supplied by power-distribution network 2
• EBS : Electronic brake controller
• SbW : Steer-by-Wire controller

Figure 1 RACE System Architecture

Integration of new subsystems (HW or SW developed as SEooC with a certain ASIL capability) in the above mentioned E/E architecture is done in 3 steps as depicted in Figure 2. Since the integration occurs during runtime, it has to be ensured that the system keeps operating according to its specification. This is especially important for such safety-critical systems, because a loss of integrity might cause damage to material and life. We call this incremental safety assessment. A once assessed set of service of a vehicle will not lose its functional safety qualities even so new services or components (sensors or actors) have been added.

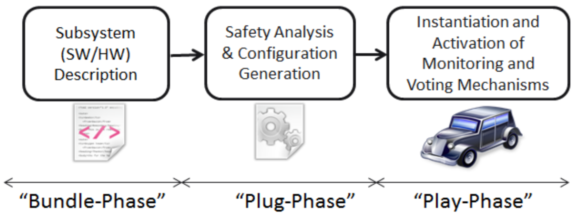
Subsystem (SW/HW) Description
Safety Analysis & Configuration Generation
Instantiation and Activation of Monitoring and Voting Mechanisms
"Bundle-Phase"
"Plug-Phase"
"Play-Phase"

Figure 2 Bundle- Phase, Plug- Phase, Play-Phase

The integration of new subsystem starts with a so called "Bundle-Phase" where the HW/SW subsystem is delivered with description information containing also the safety related information. The Plug&Play procedure is logically separated into two phases. In the "Plug-Phase" interpretation of description documents is done to re-plan the E/E configuration and adapted configuration data is provided. This is the phase where the proposed safety-assessment is performed. The next phase will just be entered in case that all required qualities especially those addressing functional safety can be ensured. In the "Play-Phase" the new configuration is enforced (Figure 2).

For the remainder of this paper, we assume such architecture as the underlying basis for the discussed methodology.

Runtime safety-assessment

Due to the fact that safety is not a modular property, meaning composition of safe systems cannot be assumed to be automatically safe; all safety quality attributes (e.g. probability of failure per hour, ASIL signal level compatibility) coming from the different subsystems/components of the system must be analyzed in order to provide safety guarantees. To do so, a safety-assessment concept that enables to model and check safety quality attributes for a system consisted of different components has to be developed. Component-based development has emerged as a promising solution for developing complex systems. The approach breaks down the system into smaller components, and the system model is obtained by composition of the individual components. This trend is the basis for our approach and enables to shift part of the safety-assessment to runtime. A quite similar approach is followed by Schneider and Trapp [6].

We believe that it is not reasonable to perform complete safety analyses (e.g., FTA, FMEA) or complex quality checks (e.g., model checking) at runtime. Therefore, it is very important to find the right level of abstraction in support of automated analysis at runtime.

In order to enable automated safety-assessment at runtime, whenever a new subsystem (HW or SW), developed as SEooC with a certain ASIL capability according to ISO 26262, will be integrated in the system a description of it and its safety requirements have to be provided to the system. The idea behind the description is to establish a predefined and standardized component description of the single (sub-) systems that are to be integrated. These descriptions can then be composed and evaluated at runtime whenever systems are to be integrated or adapted. Figure 3 depicts the information that such a description contains.

As we can see from figure 3, the subsystem description contains information about all signals (represented by data models in the MW) that are provided and required by the subsystem. In addition each signal description contains information about the ASIL level and some other safety quality requirements in the domains of value or timing. The last information is required in order to be able to evaluate the quality of the signal at runtime with a safety mechanism with corresponding ASIL level (e.g. monitoring function verifying the signal value). When a subsystem is integrated, a violation of the ASIL level compatibility for each required signal for that subsystem is checked by a Safety Manager (MW component that belongs to the above mentioned middleware and which performs the safety-assessment). In case the required level is not provided a possibility of decomposition can be evaluated.

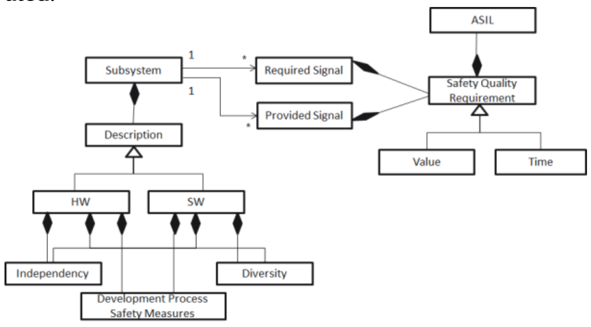

Figure 3 Subsystem and Signal Safety Requirement Description

We have examined the ASIL decomposition approach of the ISO 26262 standard [3, part 9], and how it can be correctly applied and performed at runtime. If in the system architecture, sufficiently independent and heterogeneous/diverse redundant subsystems providing the same type of signal exist, it is possible decompose the requirement of the subsystem requiring the specific signal. For example, if a subsystem to be integrated requires a certain ASIL D signal and there are two independent and diverse architectural subsystems providing that signal but with ASIL B quality, the subsystem can be integrated successfully in case a merging-functionality for the considered two signals will be available.

The Safety Manager component contains a data base of ASIL decomposition algebra patterns [3, part 9] (e.g. ASIL C = B(C) + A(C)) and a description of all elements (and the required ASIL decomposition related information) available in the system.

The Possible ASIL decomposition patterns according to ISO 26262 are:

- ASIL D = C(D) + A(D)
- ASIL D = B(D) + B(D)
- ASIL C = B(C) + A(C)
- ASIL B = A(B) + A(B)
- ASIL x = x(x) + QM(x)

The procedure of checking independency and diversity properties [3, part 9] is based on the subsystem description information.

For example, two subsystems are diverse in case they are produced by different manufacturers, use different technology and production processes, use different test methods etc. The independency property can be checked for example by checking the vendor-information of the two subsystems given in the subsystems' descriptions. For that checks such as partitions of functions or software elements, physical distance between hardware elements, common external resources are performed.

The Safety Manager Middleware component is triggered by the Plug&Play Manager during integration of a new subsystem. The Plug&Play Manager is responsible for generating the new configurations in case the new subsystem is added in the system. In case the Safety Manager approves the integration, the Plug&Play Manager executes the last steps of the integration of new components. In addition, the Safety Manager is responsible for configuration and instantiation of the required monitoring and voting mechanisms that provide the correct signal at run-time.

Implementation details

The (sub-) system description has an XML structure and as such is delivered to the middleware for all available subsystems in the system architecture. It has to be mentioned that the description to which we refer here is part of safety related description which is delivered together with the complete information of the system. The XML description is then parsed into data structures that contain all the information. For a certain system functionality (e.g. steering) a directed acyclic graph, containing the HW and SW subsystems (SEooC), is generated based on the data flow. The vertexes of the graph represent the SEooC and the directed edges the required and the provided signals. After the graph is generated the Safety Manager contains the required information in order to start the algorithm for safety assessment. Each pair of vertexes is checked regarding the compatibilities of the safety quality requirements of signals between them (required and provided signals) and if required the possibility of decomposition is evaluated as well.

A summary of the whole proposed workflow is shown in Algorithm 1.

```
Input: Subsystems subS and Service S Descriptions
Output: Safety Assesment Result

bool safetyAssesmentResult = false;
bool decompResult = false;
foreach SubSys subS in Service S do
  foreach SubSysRequiredSignal subS.reqiredSignal do
    if (findSubSystemProvidingReqSignal(subS.reqiredSignal))
      /* required signal with certain ASIL existent */
        establishLogicalRoutesBetweenSubSystems();
        instanciateConfigureMonitoringMechanisms();
        safetyAssesmentResult = true;
    else
      /* required signal with certain ASIL not existent */
      /* evaluate possibility of decomposition */
      decompResult = evaluateDecomp(subS.reqiredSignal);
      if (decompResult)
        establishLogicalRoutesBetweenSubSystems();
        instanciateConfigureMonitoringVotingMechanisms();
        safetyAssesmentResult = true;
      else
        safetyAssesmentResult = false;
        break;
  end
end
```

Algorithm 1 Safety Assessment Workflow

The algorithm that determines the possibility of decomposition is sketched below.

```
Input: Subsystems subS and Service S Descriptions, SubsystemSignalToBeDecomposed subS.requiredSignal
Output: Decomposition Evaluation Result

subSysDecriprion subS1, subS2;
bool twoSubSysProvideSignal = false;
bool decompositionRes = false;
bool independencyRes = false;
bool diversityRes = false;

/* find two subsystems providing the required signal */
twoSubSysProvideSignal =
find2SubSysProvidingSignal(subS.requiredSigna, subS1, subS2);

/* check independency and diversity of subsystems */
if (twoSubSysProvideSignal )
    independencyRes = evalIndependency(subS1.independency,
    subS2. independency);
    diversityRes = evalDiversity(subS.diversity, subS2. diversity);

    /* check if required ASIL can be achieved */
    if (independencyRes && diversityRes)
        evalDecompAlgebra(subS.requiredSigna,subS1, subS2);
        decompositionRes = true;
```

Algorithm 2 ASIL Decomposition Evaluation

4 Case Study

As outlined in the previous chapter, here we present how the previously explained concept can be used in a scenario where “steering functionality” in a vehicle based on by-wire components is to be composed from signals/data coming from different subsystems that have been developed as SEooC. The SEooC are delivered to the system together with their description information and the MW (Safety Manager) examines the possibility of their composition in order to provide the required functionality. Due to the fact that the “steering functionality” has highest ASIL level (ASIL D including required fail-operational quality as there is no mechanical backup), the software function that contains the steering algorithm is delivered as a subsystem, which is developed according to the ASIL D guidelines and which requires a sensor input signal with an ASIL D quality. However, such a signal is not provided in the system and therefore the Safety Manager starts the procedure assessing the possibility of ASIL decomposition.

Since in the system there are two sensors providing the required signal with an ASIL B quality, the MW component checks not only the independency and diversity properties but also the other safety quality requirements (e.g. the value range etc.). This means the MW checks if those subsystems don’t share common parts (e.g. connected to a different power supply in the system), are based on diverse technologies, compose the required ASIL level (Figure 4). In case all checks are positive the MW establishes the logical connections between the sensors and the software function. In addition based on the description information provided for each of the SEooC, the required voting and monitoring mechanisms (developed as safety mechanisms with a certain ASIL compatibility) providing the correct sensor value to the function during run-time are configured and instantiated.

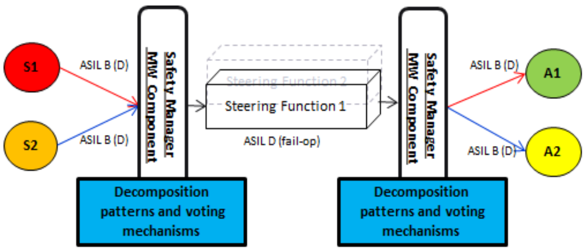
S1
S2
ASIL B (D)
ASIL B (D)
Safety Manager MW Component
Steering Function 2
Steering Function 1
ASIL D (fail-op)
Safety Manager MW Component
ASIL B (D)
ASIL B (D)
A1
A2
Decomposition patterns and voting mechanisms
Decomposition patterns and voting mechanisms

Figure 4 Safety Manager MW component performing ASIL safety-assessment and decomposition

5 Related Work

Traditionally, safety-critical systems are verified during design time using hazard analysis techniques such as Fault Tree Analysis (FTA) [10], Failure Mode and Effect Analysis (FMEA) [2], Hazard and Operability Analysis (HAZOP) etc. However in open systems as described above the need for runtime safety assessment exists. There are numerous approaches that integrate different aspects of safety assessment in the real-time systems domain. In this section we present ones that are most relevant to our work and state the differences between the approaches.

In [6] the authors share their views on safety certification perspectives for open adaptive systems, and present their approach, which is based on conditional safety certificates. Conditional certificates describe preprocessed technical safety requirements with a domain-specific language and transfer these requirements to runtime safety model. Their approach is initially oriented at assisted adaptive living or car2car communication and similar loosely coupled systems. Our approach also uses runtime safety assessment, but is focused on automotive systems and takes into consideration the recommendations from the ISO 26262 standard. In addition, the approach presented here instantiates and configures the required safety mechanisms (including monitoring and voting) that are in place to ensure the provided signal quality, rather than only working on a certificate level as in [6].

Kelly and Weaver [7] introduced the concept of Goal Structuring Notation (GSN). GSNs have been developed to provide a clear, structured, approach for modeling, developing and presenting safety arguments that are evaluated at design time in order to achieve certification. However, our approach focuses more on the question of how to describe variable properties of the subsystems to be in integrated in the automotive control system, in a form that can be evaluated at runtime. In addition, we also deal with the algorithm that performs the safety assessment.

Inverardi et al. published an approach [8] that is related to the modularity concept as in our work and to the idea of using a guarantee-demand framework (corresponding to the provided-required framework in our work). Even though, the framework aims to cope with different levels of granularity that span from code to software architecture, this approach is unspecific about the actual properties that are to be covered.

Rushby [9] examined ways in which the notion of modular certification could be interpreted in an airplane context and proposed one approach that is applicable to software components in integrated modular avionics (IMA) architectures. The approach is based on the assume-guarantee reasoning and the idea behind is that components can be certified to perform their function by using only assumptions about the behavior of other software components. In comparison to our work, this approach focuses on the modularization of safety artifacts within the development-time and does not provide a solution for formalizing the safety-related information in a way that would allow runtime evaluation.

6 Conclusion and Outlook

This paper identified the challenges regarding safety-assessment of future automotive systems. To tackle the challenges we proposed a framework and method for analyzing safety properties such as ASIL signal compatibility. Future work includes further implementation and extension of our approach (e.g. ASIL decomposition regarding SW components and their respective safety mechanisms) and integration in the "Revolution" demonstrator of the RACE project. Compatibility with existing development and certification processes is to be evaluated in future as well.

Acknowledgments

The work presented in this paper is partially funded by the German Federal Ministry of Economics and Technology under grant no. 01ME12009 through the project RACE.

Literature

[1] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann. Engineering automotive software. Proceedings of the IEEE, 95(2):356–373, 2007.

[2] M. Hillenbrand, M. Heinz, N. Adler, J. Matheis, and K. D. Mueller-Glaser, "Failure Mode and Effect Analysis based on Electric and Electronic Architectures of Vehicles to Support the Safety Lifecycle ISO/DIS 26262," in 21st IEEE/IFIP International Symposium on Rapid System Prototyping, 2010.

[3] International Organization for Standardization/Technical Committee 22 (ISO/TC 22). ISO/DIS 26262 - Road vehicles. Functional safety. Technical report, ISO, Geneva, Switzerland, November 2011.

[4] Avizienis, A., Laprie, J., Randell, B., and Landwehr, C. 2004. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Depend. Secur. Comput. 1, 11–33.

[5] S. Sommer, A. Camek, K. Becker, C. Buckl, A. Zirkler, L. Fiege, M. Armbruster, G. Spiegelberg, and A. Knoll. Race: A centralized platform computer based architecture for automotive applications. In Vehicular Electronics Conference (VEC) and the International Electric Vehicle Conference (IEVC) (VEC/IEVC 2013). IEEE, October 2013.

[6] D. Schneider and M. Trapp. A safety engineering framework for open adaptive systems. In Self Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on, pages 89–98, 2011.

[7] Kelly, T. P. and Weaver, R.. The goal structuring notation – A safety argument notation. In Proceedings of the DSN Workshop on Assurance Cases: Best Practices, Possible Outcomes, and Future Opportunities, 2004

[8] Inverardi, P., Pelliccione, P., and Tivoli, M.. Towards an assume-guarantee theory for adaptable systems.In Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS'09). 2009, pages106–115.

[9] Rushby, J., Modular certification. NASA contractor rep. CR-2002-212130, NASA Langley Research Center., 2002

[10] Ericson, C.A., Fault tree analysis: a history. In: Proceedings of the 17th international system safety conference, 1999

[11] M. Eichhorn, M. Pfannenstein, D. Muhra, and E. Steinbach. A soa-based middleware concept for in-vehicle service discovery and device integration. In Intelligent Vehicles Symposium (IV), 2010 IEEE, pages 663–669, 2010.

[12] G. Gehlen, E. Weiss, and A. Quadt. Service oriented middleware for automotive applications and car maintenance. In Proceedings of the 1nd Workshop on Wireless Vehicular Communications and Services for Breakdown Support and Car Maintenance, pages 42–46, Nicosia, Cyprus, Apr 2005. RWTH Aachen University.