# Supporting Cyber-Security Based on Hardware-Software Interface Definition

Georg Macher[1(✉)], Harald Sporer[2], Eugen Brenner[3], and Christian Kreiner[3]

[1] AVL List GmbH, Graz, Austria
georg.macher@avl.com
[2] pewag International GmbH, Graz, Austria
[3] Institute for Technical Informatics,
Graz University of Technology, Graz, Austria
{brenner,christian.kreiner}@tugraz.at

**Abstract.** The automotive industry has an annual increase rate of software implemented functions of about 30 %. In the automotive domain the increasing complexity of systems became challenging with consumer demands for advanced driving assistance systems and automated driving functionalities, and the thus broadening societal sensitivity for security and safety concerns, such as remote control of cars by hacking their IT infrastructure.

As vehicle providers gear up for the cyber-security challenges, they can leverage experiences from many other domains, but nevertheless have to face several unique challenges. The recently released SAE J3061 guidebook for cyber-physical vehicle systems provides high-level principles for automotive organizations to identify and assess cyber-security threats and design cyber-security aware systems in close relation to ISO 26262. Although functional safety and cyber-security engineering have a considerable overlap regarding many facets, such as analysis methods and system function thinking, the definition of system borders (item definition vs. trust boundaries) often differs largely. Therefore, appropriate systematic approaches to support the identification of trust boundaries and attack vectors for the safety- and cybersecurity-relates aspects of complex automotive systems are essential. In the course of this paper, we analyze a method to identify attack vectors on complex systems via signal interfaces. We focus on a central development artifact of the ISO 26262 functional safety development process, the hardware-software interface (HSI), and propose an extension for the HSI to support the cyber-security engineering process.

**Keywords:** ISO 26262 · SAE J3061 · Automotive systems · Hardware-software interfaces · Cyber-security · Functional safety

## 1 Introduction

The emergence of embedded automotive systems over the last decades has affected the development of vehicles, promising to improve the safety of drivers


C. Kreiner et al. (Eds.): EuroSPI 2016, CCIS 633, pp. 148–159, 2016.
DOI: 10.1007/978-3-319-44817-6_12

and support new applications. Exploiting the rising vehicle-to-vehicle and vehicle-to-infrastructure paradigms (growing to over 210 million Euros by 2016), future vehicles will have multiple inter-vehicle connections as well as capabilities for (wireless) networking with other vehicles and non-vehicle entities such as charging stations and traffic lights [1].

The resulting inter-connectivity increases attack surfaces and their damage potential, especially in light of the estimation that, worldwide, over a million people fall victim to cyber-crime every day, with a global cost of cyber-crime valued at 313 billion Euros in 2011 [2]. Embedded automotive system technologies offer great benefits, but they also bring new risks for users today, becoming critical for both quality and security performances.

Before the introduction of wireless connections and automated driving functionalities, vehicles were physically isolated machines with mechanical controls. Extra-functional properties of concern were mainly timing, reliability and functional safety. Today the automotive domain is focusing on adapting established functional safety processes and methods for security engineering (e.g. the recently available SAE J3061 [3]). Although functional safety and cyber-security engineering have a considerable overlap regarding many facets, the elements of concern are not identical in the two engineering disciplines. One example is the identification of trust boundaries for the safety- or cyber-security-related aspects of complex automotive systems and the definition of system borders in ISO 26262 context (item definition). Thus, appropriate systematic approaches to support the identification of trust boundaries are essential.

In the course of this paper, we analyze a way to identify trust boundaries and attack vectors on complex systems via signal interfaces based on the hardware-software interface (HSI), a central development artifact of the ISO 26262 functional safety development process. Furthermore, we propose an extension for the HSI to support the cyber-security engineering process.

The paper is organized as follows: Sect. 2 presents an overview of related works. In Sect. 3 a description of the proposed approach and detailed information about the individual items is provided. A brief evaluation of the approach is presented in Sect. 4. Finally, Sect. 5 concludes with an overview of the approach presented.

## 2 Related Work

An unambiguous definition of the hardware-software interface has become vital in the context of the road vehicles functional safety norm ISO 26262 [4]. But neither the functional safety standard nor automotive process reference model of Automotive SPICE [5] prescribe a specific methodology for the development of this artifact. Although an unambiguous specification of the various signals of embedded automotive systems to define the hardware/software interface is of high importance for the automotive domain, publications on HSI definitions are rare.

In the automotive domain hardware and software development cycle times differ significantly in length and software development is typically separated into

several abstraction layers (such as application software (ASW), microcontroller abstraction layers (MCAL), basic functionality drivers (BSW)). This approach excludes hardware specific details and enables the establishment of focused software development teams (e.g., basic software developer, application software engineers, software integrators), but on the other hand it sometimes obfuscates the importance of HSI development.

In [6] a model-based development (MBD) approach for an ISO 26262 aligned HSI definition is presented. This work combines spreadsheet tools (such as Excel) and MBD tools in a bidirectional manner to enable a tool-independent method of engineering HSI definitions with spreadsheet tools and transformation of the generated information into a reusable and version-able model representation.

A domain-specific modeling approach for mechatronic systems with an integrated HSI definition feature is presented in [7]. The approach of this work has mainly been created for the development of embedded mechatronic based electric/electronic systems (E/E systems) in the automotive field and is based on a domain-specific language tailored for the specific needs of domain experts. The focus of this work was particularly set to simplify the work of domain experts who disfavor system modeling approaches (like UML or SysML). Other works postulate the problematic of defining HW/SW interfaces in the context of System on Chip (SoC) development [8], or are part of an emerging domain-independent paradigm for contract-based design. The contracts specify the input and output behavior of a component and provide a guaranteed behavior [9]. Such an approach can be used for software component safety contracts [10] as well as contract-based embedded system development [11,12]. Nevertheless, these approaches are not yet very common in the automotive domain.

### 2.1 HSI Relation to Automotive SPICE

The Automotive **S**oftware **P**rocess **I**mprovement and **C**apability D**e**termination reference model [5] is based on the international standard ISO 15504 [13] and is primarily used in Europe, as well as in some parts of Eastern Asia. The reference model does not specify how processes have to be implemented. Instead, desired process outcomes are defined and described in more detail by best practice (BP) characterization (base or generic practices). The model does not address the demand for a hardware-software interface directly, but some guidance on HSI specification can be extracted from general interface topics of the system engineering processes (SYS) and software engineering processes (SWE).

- SYS.3.BP3, stipulates the definition (identify, develop, and document) of system element interfaces, which are equivalent to the hardware software interface.
- SYS.3.BP4 regards the description of the dynamic behavior of and between the system elements, which have to be taken into account in the HSI definition as well.
- SYS.4.BP3 postulates that the system integration test needs to provide evidence of consistency between the interfaces and the architectural design, which relates to the HSI definition.

- SWE.2.BP3 and SWE.2.BP4 can be interpreted in a similar way to their system level counterparts (SYS.3.BP3, SYS.3.BP4).
- SWE.2.BP5 regards the determination and documentation of the resource consumption objectives of all relevant software architectural design elements; to support this, the HSI definition shall include information on resource consumption.
- SWE.3.BP2, SWE.3.BP3 and SWE.3.BP4 can be interpreted in a similar way to their SW architecture counterparts (SWE.2.BP3, SWE.2.BP4 and SWE.2.BP5); nevertheless, signals communicated between the components on the most detailed software level do not directly belong to the HSI.
- SWE.5.BP3 requires a description of the interaction between relevant software units and their dynamic behavior, which can be interpreted in a similar way to its system level counterpart (SYS.4.BP3).

### 2.2 HSI Relation to ISO 26262

The HSI definition is one of the most important and essential work-products among the many required by ISO 26262. The HSI specifies the hardware and software interactions in consistency with the technical safety concept, which includes hardware components that are controlled by software and support the software execution.

The HSI document is the last development artifact of the system development phase and the starting point for parallel development of hardware and software. The HSI definition thus requires mutual domain knowledge of hardware and software and is usually the result of a collective workshop of hardware, software, and system experts. The HSI is the linkage between different levels of development and is used to align topics relevant to both hardware and software development. Furthermore, the HSI shall be continuously refined in the hardware and software product development phases, which are described in Parts 5 and 6 of the ISO 26262.

Although many best practice articles and books related to ISO 26262 have been published, the hardware-software interface has rarely been highlighted in any of these publications. This might be caused by the fact that HSI definition requires mutual domain knowledge and the responsibility for this artifact differs from company to company.

The majority of information concerning how to specify the interface in relation to functional safety can be found in Clause 7.4.6 of Part 4 of the standard. Additionally, the informative Annex B of Part 4 of ISO 26262 provides information concerning the possible content of the interface definition.

## 3 Hardware-Software Interface Definition with Security Extension

As mentioned previously, the HSI definition is probably the most crucial and essential work-product required by ISO 26262 related development approaches.

It requires mutual knowledge of hardware and software components and their interactions and needs refinement by these two development processes after initial establishment at the end of the system development phase. By now, the HSI specification no longer consists of only a single spreadsheet description of all signals from hardware to software and vice versa. Supplementary information, such as resource consumption objectives, HW specifics, and controller module configurations also need to be considered for an ISO 26262 or AutomotiveSPICE compliant HSI definition.

Table 1 itemizes a list of essential HSI attributes extracted from standards (ISO 26262 [4], AutomotiveSPICE [5], and SAE J3061 [3]), scientific papers [7,14], and the authors' experiences. Additionally, information has been added (marked with black boxes) to support security related identification of attack vectors on complex systems via their signal interfaces. The highlighted information can be used to identify attack surfaces and establish a defense in depth security pattern for specific signals.

To that aim, signals that are safety or cyber-security relevant inherit their ASIL from the hazard analysis and risk assessment (HARA; requested by ISO 26262) and/or their security level from threat analysis and risk assessment (TARA; requested by SAE J3061). Depending on the related security level / ASIL the signal shall be protected against cyber-security attacks according to a defense in depth pattern focusing on the (simplified) OSI model [16]. The OSI model is a conceptual model that partitions a communication system into abstraction layers (in the original version seven layers). Its goal is the interoperability of diverse communication systems with standard protocols. A layer serves the layer above it and is served by the layer below it.

In addition, the idea behind the defense in depth approach is to defend a system against any particular attack using several independent methods. If any of the layers fails to protect, then the next layer is in place to provide protection. Such a defense in depth approach is also proposed for automotive systems in general by [17,18] and for in-vehicle infotainment systems in particular by [19]. Another defense in depth approach for securing Ethernet communication for autonomous driving is presented in [20]. This work claims generally that enhanced connectivity and the dynamics of the security threats demand the establishment of several security barriers in order to avoid full exposure in case a security mechanism is bypassed.

Moreover, enhancing the HSI definition with supplementary cyber-security information and related signals helps to determine trust boundaries and attack vectors by focusing on signals and thus identifying controllers which can intervene with the involved signals. To that aim, all signals required for the system are analyzed and based on this analysis all control units, which have access to these signals are identified. These control units are within the same trust boundary. Systems within the same trust boundary are equally trusted. Access to the trust boundary is able only via dedicated devices (gateways) which have connections outside the trust boundaries. Thus, it is required that gateways prevent the misuse of trust and protect the control units within a trust boundary

**Table 1.** Essential HSI attributes, comments and origin

| Layer | Attribute | Comment | Origin |
|---|---|---|---|
| conceptual | signal name | significant name | [14] |
| | signal description | short signal description | ISO 26262 Part 6 |
| | signal direction | input or output | [14] |
| | signal source/sink | actuator or sensor related to signal | [7] |
| | ASIL | Automotive Safety Integrity Level | ISO 26262 Part 4 |
| | Security Level (SecL) | security metric | SAE J3061, [15] |
| physical | supply voltage | - | [7] |
| | physical min value | - | ASPICE |
| | physical max value | - | ASPICE |
| | physical unit | - | ISO 26262 Part 6,ASPICE |
| | accuracy | % range of value | ISO 26262 Part 6 |
| | HW interface type | digital, analog, bus ... | ISO 26262 Part 6, ASPICE |
| | HW pin | pin number or identifier | ISO 26262 Part 5 |
| data | message ID | for bus communications | [7] |
| | message offset | | [7] |
| | cycle time internal | xCU internal refresh rate | ISO 26262 Part 6, ASPICE |
| | cycle time external | cycle time of digital signal from external | [7] |
| | trigger | identifier of trigger | ISO 26262 Part 6 |
| | operation mode | information if any special operation mode required | ISO 26262 |
| | HW diagnostic feature | diagnostic feature description | ISO 26262 |
| | memory type | - | ISO 26262 |
| | data protection | special security information | ASPICE |
| | timing dependencies and sequence order | - | ASPICE |
| presentation | SW signal name | signal identifier for ASW | [14] |
| | initial value | - | [7] |
| | SW data type | - | ASPICE |
| | scaling LSB | fixed-point arithmetic scaling | [14] |
| | scaling offset | | |
| | SW min value | - | ASPICE |
| | SW max value | - | ASPICE |
| | SW accuracy | % range of value | ISO 26262 |
| | SW unit | physical unit representation | ASPICE |
| | default value | default value in case of invalid signal | [14] |
| | detection time | time to fault diagnosis | ISO 26262 |
| | reaction time | reaction time after fault detection | ISO 26262 |

from attacks. This identification of trust boundaries and gateways which protect the boundaries is both crucial and cumbersome for complex system and network structures. Therefore, using the HSI definition provides a structured and methodical pattern for identification.

## 4 Application of the Proposed Approach

This section demonstrates the application of the presented approach for a representative safety and security relevant automotive embedded system. To that aim a basic concept of an electronic steering column lock (ESCL) has been chosen. This use-case is an illustrative example, reduced for training purposes and is not intended to be exhaustive or representing leading-edge technology or solutions.

Figure 1 shows a block diagram depiction of the use-case from a safety perspective (item definition). The depiction shows all involved components of an ESCL from sensors to actuators. As can be seen, the actuator is simply an electric motor which moves the bolt, controlled by a motor controller and an electric control unit (ECU). The required sensor signals are (a) a feedback channel of the bolt position (represented via endpos signal), (b) power supply and ignition key status information (CL30 and CL15), and (c) vehicle status information via CAN bus (ignition key status, vehicle speed signal, gear lever position).

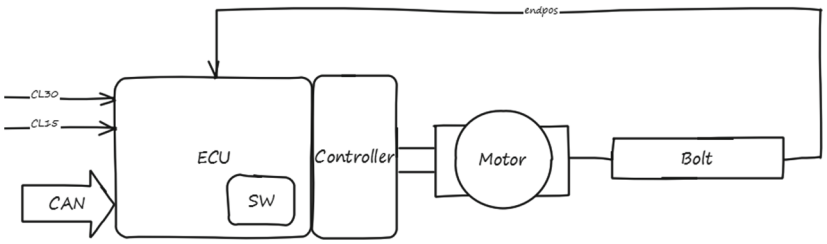


**Fig. 1.** Block diagram depiction of use-case (Item definition)

An excerpt of an ESCL HARA and TARA analysis [21] (using a combined approach by applying the SAHARA method [15]) reveals the security threat '*SH_1: spoofing of key-less-go off signal, vehicle speed 0kmph and gear lever in park position*', resulting in $SecL = 2$ (security level) and security goal '*prevent spoofing of steering column lock signals*', and is also related to safety goal '*SG1: prevent unwanted steering column locking*' with $ASIL\ D$.

The use-case example represents a safety-critical and cyber-security related component and requires the application of an ISO 26262 aligned development process. Complying with ISO 26262 aligned development process requirements, the HSI artifacts for the ESCL system depicted in Fig. 2 are generated at the end of the system development phase.

| | HSI | | ESCL | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | Signal name | | | CL30 | CL15 | endpos | ignition key status | vehicle speed signal | gear lever position |
| 3 | signal description | | common | supply voltage signal from battery | ignition-starter switch signal | end position signal of ESCL bolt | ignition-starter switch signal | actual vehicle speed | actual gear lever position |
| 4 | Sensor/Actuator | | | CL30 | CL15 | endpos1 | -- | -- | -- |
| 5 | Direction | | | in | in | in | in | in | in |
| 6 | ASIL | | | ASIL B | ASIL B(D) | ASIL B | ASIL B(D) | ASIL B(D) | ASIL B(D) |
| 7 | SecL | | | 0 | 0 | 0 | 2 | 2 | 2 |
| 8 | Source(CAN/ANA/DIG) | | | ANA | ANA | DIG | CAN | CAN | CAN |
| 9 | physical unit | | physical | V | V | -- | -- | -- | -- |
| 10 | physical range lower limit | | | 0 | 0 | 0 | -- | -- | -- |
| 11 | physical range upper limit | | | 12 | 12 | 12 | -- | -- | -- |
| 12 | supply voltage | | | 12 | 12 | 12 | -- | -- | -- |
| 13 | signal tolerance | % | | 5 | 5 | -- | -- | 1 | -- |
| 14 | interface | | | analog in | analog in | digital in | CAN A | CAN A | CAN A |
| 15 | pin | | | Port A11 | Port A12 | Port A15 | Port B12 | Port B12 | Port B12 |
| 16 | refresh rate | ms | data | -- | -- | -- | 100 | 100 | 100 |
| 17 | cycle time | ms | | 10 | 10 | 10 | 10 | 10 | 10 |
| 18 | message ID | | | -- | -- | -- | 0x100 | 0x100 | 0x100 |
| 19 | message offset | | | -- | -- | -- | 0 | 0 | 0 |
| 20 | trigger | | | -- | -- | -- | | | |
| 21 | operation mode | | | normal | normal | normal | normal | normal | normal |
| 22 | HW diagnostic | | | ref. Voltage | ref. Voltage | -- | CRC | CRC | CRC |
| 23 | register-type | | | RAM | RAM | RAM | RAM | RAM | RAM |
| 24 | data protection | | | -- | -- | -- | -- | -- | -- |
| 25 | dependency | | | -- | -- | -- | -- | -- | -- |

**Fig. 2.** Excerpt of the HSI definition of the ESCL use-case

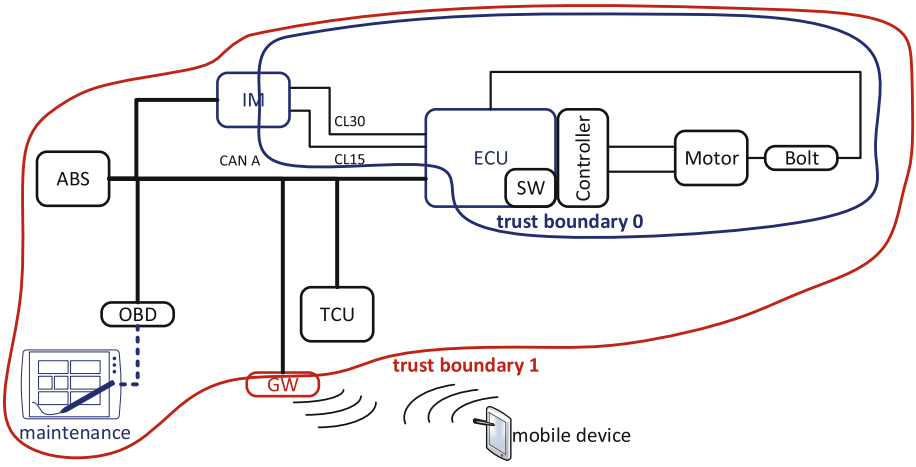


**Fig. 3.** Trust-boundary layers of use-case

As can be seen in Fig. 2, the *SecL* of the three directly connected signals (endpos, CL30, and CL15) are treated as 0 (not security relevant). This due to the fact that in order to mount a security attack, these signals would have to be manipulated in the vehicle directly at the ESCL system and that these signals are within the same trust boundary 0 (see Fig. 3). On the other hand, the three signals provided via CAN bus (thus provided from outside of trust boundary 0 of Fig. 3) are assigned a $SecL = 2$. This *SecL* indicates a possible cyber-security vulnerability and thus requires built-in security solutions exhibiting a defense-in-depth approach.

The realization of these protections in vehicle systems requires the coordinated design of multiple security technologies (such as isolation of safety critical systems, secure boot, tamper protection, message authentication, network encryption and many others). More details on these automotive security technology best practices, and how and which to choose for different security levels are out of the scope of this work, but can be found, among others, in [17,19]. Currently no standardization for the coordination of security designs has been established and it is up to the manufacturers to decide how to provide a secure context. Thus, we established the following design guideline for signal security for the different security levels (based on [22]):

| Layer | Attribute |
|---|---|
| SecL = 0 | no additional requirements |
| SecL = 1 | verify origin of message |
| | verify integrity of message |
| SecL = 2 | check volumes of messages |
| | detect abnormal behavior |
| | immutable device identification |
| | intrusion detection |
| SecL = 3 | encrypted communication |
| | data encryption |
| SecL = 4 | establishing of private communication channel |
| | correct cycle detection |
| | blocking of unapproved and inappropriate messages |

For the example, the three CAN signals required by ESCL (assigned $SecL = 2$) have to be verified by the origin of message (this requires an immutable device identification) and message integrity (e.g. CANs CRC). Also, a detection of abnormal behavior of the CAN bus including a check of message repeat rate and intrusion detection is required.

Based on the HSI identification of the interfaces providing the signals (see Fig. 2 - line 15), devices connected to this interface can be easily identified and trust-boundaries for the specific system identified. This enables a complete

identification of involved controllers for a further analysis of interfaces and the establishment of barriers for cyber-security attacks. To do this, we started with the ISO 26262 item of the ESCL system (Fig. 1) and filtered the content of the HSI for signals related to the ESCL system (Fig. 2). In the first step we identified controllers which have access to these signals. These controllers (depicted in Fig. 3) either generate the signals directly (such as the vehicle immobilizer (IM)) or are connected to the same communication bus (antilock braking system (ABS), on-board diagnosis connector (OBD), transmission control unit (TCU) and wireless gateway (GW)). For the second step we identified the inner trust boundary 0 which includes signals directly connected to the electric control unit (ECU). Simultaneously, gateways to the trust boundary 0 are identified (IM and ECU), which are required to ensure protection of the integrity of the trust boundary 0. In the next step the trust boundary of the remaining signals is established. Figure 3 shows a depiction of the first trust-boundary layers of the use-case. As can be seen in the depiction, trust boundary 1, which covers the first layer of all signals related to the ESCL system, also includes the wireless gateway (GW), which appears as a gateway to trust-boundary 1 and therefore enables remote cyber-security attacks on the ESCL. Additionally, if the on-board diagnostic connector (OBD) does not provide protection mechanisms for trust-boundary 1 (usually the case in common vehicle designs), maintenance systems are included in trust boundary 1 as well; a fact that could be easily overlooked and enabled security attacks recently described in [23,24].

## 5 Conclusion

Vehicle manufacturers are currently gearing up for the newly arising cyber-security challenges. Although they can leverage experiences from many other domains, nevertheless they have to face several unique challenges. Security standards do not need to be created from scratch for the automotive domain and are frequently strongly related to the safety processes.

Functional safety and cyber-security engineering have an overlap regarding many facets, but some development artifacts (such as the definition of system borders (item definition vs. trust boundaries)) often differ completely. To that aim, we have proposed a way to identify trust-boundaries and security design guidelines for the signal security of complex systems via signal interfaces defined in the hardware-software interface (HSI) definition. The application of the approach presented has been demonstrated based on a representative safety and security relevant use-case, an electronic steering column lock (ESCL). Although this use-case is intended for training purposes and represents neither an exhaustive nor a commercially sensitive project, the main benefits of the approach have been made evident.

**Acknowledgments.** This work is supported by the $EMC^2$ project. The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement nr 621429 (project $EMC^2$) and from the Austrian Ministry for Transport, Innovation and Technology (BMVIT) in the Program IKT der Zukunft under FFG grant agreement nr 842537.

## References


1. Bisson, P., Martinelli, F., Granadino, R.R.: Cybersecurity Strategic Research Agenda-SRA. In: European Network and Information Security (NIS) Platform NISP-Working Group, 3 (WG3), vol. v0.96, pp. 1–201, August 2015
2. Cercone, M., Ernst, T.: An EU cybercrime centre to fight online criminals and protect e-consumers. European Commission-Press release, March 2012
3. Vehicle Electrical System Security Committee, SAE J3061 Cybersecurity Guidebook for Cyber-Physical Automotive Systems
4. ISO-International Organization for Standardization, ISO 26262 Road vehicles Functional Safety Part 1–10 (2011)
5. The SPICE User Group, Automotive SPICE Process Assessment/Reference Model V3.0, July 2015
6. Macher, G., Sporer, H., Armengaud, E., Kreiner, C.: A versatile approach for ISO26262 compliant hardware-software interface definition with model-based development. SAE Technical Paper, SAE International (2015)
7. Sporer, H., Macher, G., Kreiner, C., Brenner, E.: Resilient interface design for safety-critical embedded automotive software. In: Zizka, J., et al., (eds.) Sixth International Conference on Computer Science and Information Technology, CCSIT 2016, Zurich, Switzerland, pp. 183–199. Academy and Industry Research Collaboration Center (AIRCC) (2016)
8. King, M., Dave, N., Arvind: Automatic generation of hardware/software interfaces. In: Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS XVII, New York, NY, USA, pp. 325–336. ACM (2012)
9. Cimatti, A., Tonetta, S.: A property-based proof system for contract-based design. In: 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 21–28, September 2012
10. Soderberg, A., Johansson, R.: Safety contract based design of software components. In: 2013 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 365–370, November 2013
11. Damm, W., Hungar, H., Josko, B., Peikenkamp, T., Stierand, I.: Using contract-based component specifications for virtual integration testing and architecture design. In: Design Automation Test in Europe Conference Exhibition (DATE) 2011, pp. 1–6 (2011)
12. Iber, J., Höller, A., Rauter, T., Kreiner, C.: Towards a generic modeling language for contract-based design. In: 2015 Workshop Proceedings 2nd International Workshop on Model-Driven Engineering for Component-Based Software Systems (ModComp), p. 24 (2015)
13. ISO-International Organization for Standardization, ISO/IEC 33000 Series on Process Assessment (2014)
14. Macher, G., Sporer, H., Armengaud, E., Brenner, E., Kreiner, C.: Using model-based Development for ISO26262 aligned HSI Definition. In: EDCC Conference Proceedings (2015)
15. Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C.: SAHARA: a security-aware hazard and risk analysis method. In: Design Automation Test in Europe Conference Exhibition (DATE) 2015, pp. 621–624 (2015)
16. ISO-International Organization for Standardization, ISO IEC 7498–1 Information technology-Open Systems Interconnection-Basic Reference Model: The Basic Model (1994)

17. Brown, D., Cooper, G., Gilvarry, I., Rajan, A., Tatourian, A., Venugopalan, R., Wheeler, D., Zhao, M.: Automotive Security Best Practices, White Paper, pp. 1–17 (2015)
18. Hahn, T., Matthews, S., Wood, L., Cohn, J., Regev, S., Fletcher, J., Libow, E., Poulin, C., Ohnishi, K.: IBM Point of View: Internet of Things Security, White paper, April 2015
19. Windriver, Improving Android Security for Automotive with a Defense-In-Depth Strategy, White Paper (2013)
20. Pallierer, R., Ziehensack, M.: Secure Ethernet Communication for Autonomous Driving, February 2016
21. Macher, G., Riel, A., Kreiner, C.: Integrating HARA and TARA-How does this fit with Assumptions of the SAE J3061, Software Quality Professional (2016)
22. Otsuka, S., Ishigooka, T., Oishi, Y., Sasazawa, K.: CAN Security; Coste-Effective Intrusion Detection for Real-Time Control Systems, SAE Technical Paper 2014–01-0340 (2014)
23. Greenberg, A.: Hackers cut a Corvette's brakes via a common car gadget, November 2015
24. Mahaffey, K.: Hacking a Tesla Model S: What we found and what we learned, August 2015