Noname manuscript No.
(will be inserted by the editor)

A Tour of Secure Software Engineering Solutions for Connected Vehicles

Antonia Bertolino · Antonello Calabro' · Felicita Di Giandomenico · Giuseppe Lami · Francesca Lonetti · Eda Marchetti · Fabio Martinelli · Ilaria Matteucci · Paolo Mori

Received: date / Accepted: date

Abstract The growing number of vehicles daily moving on roads increases the need of protecting the safety and security of passengers, pedestrians, and vehicles themselves. This need is intensified when considering the pervasive introduction of Information and Communication Technologies (ICT) systems into modern vehicles, because this makes such vehicles potentially vulnerable from the point of view of security.

The convergence of safety and security requirements is one of the main outstanding research challenges in software-intensive systems. This work reviews existing methodologies and solutions addressing security issues in the automotive domain with a focus on the integration between safety and security aspects. In particular, we identify the main security issues with vehicular communication technologies and existing gaps between state-of-art methodologies and their implementation in the real world. Starting from a literature survey and referring to widely accepted standards of the domain, such as AUTOSAR and ISO 26262, we discuss research challenges and set baselines for a holistic secure-by-design approach targeting safety and security aspects all along the different phases of the development process of automotive software.

A. Bertolino and A. Calabro' and F. Di Giandomenico and G. Lami and F. Lonetti and E. Marchetti
Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"
Consiglio Nazionale delle Ricerche
via G. Moruzzi 1 - 56124 Pisa, Italy
E-mail: {firstname.lastname}@isti.cnr.it

F. Martinelli and I. Matteucci and P. Mori
Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
via G. Moruzzi, 1 - 56124 Pisa, Italy
E-mail: {firstname.lastname}@iit.cnr.it

1 Introduction

The usage of Information and Communication Technologies (ICT) in automotive is increasing over and over. Indeed, the user perception of *connected vehicle ecosystems* is evolving from the concept of a niche product to a must-have feature. Via cellular-based communication or Wi-Fi connection, a vehicle is currently capable of gathering a large set of information, such as weather, road conditions, accidents, traffic conditions, or other comfort features (e.g., AUDI Connect, BMW ConnectDrive) directly from the Internet or proprietary (Ad Hoc) networks related to the trip. At the same time, vehicles can send back a similar set of information to vehicle tracking systems in order to either enhance cargo and driver safety or optimize fleet resources.

This trend calls attention to cyber-security issues related to the many cars, trucks, and motorbikes that interact with each other while moving on roads. Security features, such as data confidentiality, and integrity features, such as availability of the connectivity functionality, must be guaranteed concerning both the information exchanged among vehicles, and between the user mobile device and the connectivity system of the vehicle. In summary, automotive systems must provide robust mechanisms preventing the exploitation of vulnerabilities by malicious agents. However, compared to functional safety, the practical development of secure systems is still less structured and standardized. In most cases, and especially for business software and non-defence systems, security is not really integrated in software engineering processes, in spite of the fact that the development of secure software and security assessment have been under study for several decades. Researchers have investigated several solutions targeting security requirements in the different stages of the software development process of embedded systems, see, for instance, [8, 19, 75, 71]. In this paper, we propose a review of solutions dealing with the assurance of security features in the automotive domain, pointing out existing issues and identifying research challenges in the area. We also propose a *Security-by-Design* approach in order to reduce the gap between needs and solutions in the automotive domain. Secure by design, in software engineering, means that the software has been designed from scratch in such a way to accomplish security requirements.

The paper is organized as follows: Section 2 presents the motivations of this work by recalling real world examples of security breaches in the automotive domain. Section 3 surveys cyber-security attacks and issues in different automotive scenarios. In particular, according to the involved parties in the communication, four scenarios are identified: Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2X), Intra Vehicle (IntraV), and User to Vehicle (U2V). Section 4 reviews existing solutions of each step of the software development process. Section 5 points out several research challenges on secure software engineering for the automotive sector and proposes a Secure-by-Design toolchain to overcome (some of) them. Finally, Section 6 draws the conclusion.

2 Motivation

In 2013, more than 1 billion sensors were sold to the automotive industry, doubling 2009 levels, and embedded connectivity solutions began appearing in 2014[1]. As a consequence, serious concerns arise regarding, among others, privacy, safety, and security of automotive ecosystems, and related standards. A connected vehicle can track the overall behaviour of the driver, including information about location, driving style and additional trip parameters, such as fuel economy, and more sensitive parameters from the privacy point of view, such as the phonebook of a mobile phone, information about home location, data related to the homelink remote control.

Without complete, efficient, and robust security control systems, a concrete risk exists that such private information could be accessed by third parties, e.g., interested to push offers or to profile the user. Thus, security has become a serious issue with connected vehicles. The apocalyptic scenario described by Stephen King in the movie "Maximum Overdrive", where self-driving trucks threaten societal security by creating traffic jams, injuring people, and stalling circulation and basic transportation services, is not far from reality. Several studies have investigated the cyber-crime through automotive networks and the risks related to losing control of autonomous vehicles. As emerged from these studies, a series of potential attacks involving intra/inter vehicle systems communications can be conceived, spanning from a "simple" traffic jam to more complex and dangerous attacks. In real world, cyber-security attacks have been perpetrated to vehicles as a demonstration of the vulnerabilities of connected vehicles, such as:

- In 2010, some security researchers showed how to "kill"a car engine remotely, i.e., make the car engine exploitable, by turning off the brakes so that the vehicle would not stop, and making instruments give false readings. This attack highlighted that the computer systems used to control modern cars are vulnerable, "fragile"and easy to subvert[2].
- In July 2015, the WIRED magazine reported the news of an hijacking perpetrated to a Jeep Cherokee[3]. Indeed, "two hackers remotely toyed with the air-conditioning, radio, and wind-shield wipers [...]"and, finally, cut the engine. The hackers exploited continuous internet-connection of the U-Connect infotainment system of the vehicle, through which the car manufacturer can update the software over-the-air. The lack of security controls of the U-Connect system allowed the uploading of a malicious version of the software, through which remote control of the vehicle was made possible. This fact was published just a few days after the issuing of a fine of $70M that Fiat Chrysler Automobiles (FCA) agreed to pay the National Highway Traffic Safety Administration (NHTSA) related to the management of recall campaigns of vehicles in the United States.

[1] https://goo.gl/jeHxIg
[2] http://goo.gl/46ojKC
[3] http://goo.gl/fAUfBv

– After less than ten days, also General Motors (GM) has been target of a carjacking[4]. A 29-year-old software developer figured out a way to attack GM cars by using a personally revised version of OnStar, which is a system built into many GM cars that lets owners do things like remotely unlock or start their cars from an app or a phone service. The tampered version, called OwnStar, allowed the hacker to locate, un-lock, and start a GM vehicle by simply attaching a device somewhere on the targeted car. Whenever the car owner opens the OnStar mobile app within WiFi range of the vehicle, the OwnStar gadget placed on the car discloses all kinds of valuable information to the hacker.

These examples of threats could be the first steps towards the realization of some possible catastrophic scenarios, where cyber-security attacks on automotive systems could be exploited for creating dangerous and risky situations for human life and hence for the whole society. Not far from reality could be, for example, the case in which, by managing the Electronic Control Unit (ECU), a whole set of vehicles is hacked so that the hypothetical attacker could remotely stop the vehicles altogether by simulating an emergency condition, and thus paralyze important highway connections.

The threat of vehicle hacking cannot be underestimated. To face cyber security attacks, specific solutions to detect anomalies in the automotive system and to recover from them are needed. In the following, starting from an overview of the main security threats in automotive, we provide a review of existing protection mechanisms and point out some research challenges to both enhance existing solutions and propose new ones able to cope with still open or new security issues.

3 Automotive cyber security attacks

The nature of attacks to connected vehicles could be various. The attacker could be even the car owners themselves, who, for some reason, might want to alter the configuration of their own car, and involuntarily compromise the overall car safety. Indeed this scenario is realistic considering the different guidelines available on the Internet [67] on how to hack a car. As reported in [41], the reasons for hacking a car are disparate: i) Data theft (several kinds of data accessible from the car); ii) Extortion; iii) Fraud and deception (altering or deleting schedule log and records); iv) Freight and goods theft; v) Automotive Hactivism (cyber-infiltration of a vehicle system that is politically or ideologically-motivated); vi) Immobilisation; vii) Mischief and malevolence (individual hackers testing defences and their skills or wanting to inflict damage and/or disruption out of spite); viii) Premises security and burglary (vehicle data that reveal businesses and homes are unoccupied); ix) Industrial espionage and illegal access to intellectual property; x) Infliction of political or damage to reputation; xi) Script kiddies adversarial hackers pitting their

[4] http://goo.gl/aXaWzl

skills against the automotive software safeguards; xii) Sabotage or degrading of vehicle and connected system performance; xiii) Terrorism disabling vehicles as part of an attack; and xiv) Vehicle identification re-assignment (for stolen cars).

Protecting vehicle against unauthorized accesses and manipulation is a central challenge for current and future ECUs. Automotive Original Equipment Manufacturer (OEM) and suppliers have already been working for a number of years in this area.

In recent years, several research activities have been carried out in the field of automotive cyber-security, in order to define a detailed list of threats, and a set of recommendations to car manufacturers to overcome them, mostly related to:

Authenticity: data exchange between senders and receivers is trustworthy. Indeed, information exchanged between, e.g., the user mobile device and the vehicle, is confidential (user contacts, phone numbers, etc.) and must not be stolen by an untrusted party. The system must provide mechanisms to authenticate the parties that exchange information with both other vehicles and the road infrastructure.

Integrity: information contents are complete and unmodified. For instance, the mobile applications interacting with the vehicle system must be evaluated as trustful by ad-hoc mechanisms of the vehicle system itself. The same for the information (requests, data, etc.) exchanged between the mobile device and the connectivity systems in order to avoid data misinterpretation and failure in control units. Otherwise, hackers could exploit these vulnerabilities to get sensitive vehicle data or to acquire control of parts of the car.

Confidentiality: data is encrypted and can only be read by authorized nodes. The application of remote controls from, e.g., a mobile device, should be able to manage only those functionalities for which they have been developed (for example, safe control units on the vehicle should not be remotely controllable). Besides, users of remote controls should be authenticated (for example, the seat of the driver must not be controlled by another person without the agreement of the driver).

In [53], Koscher et al. experimentally evaluate the security features of a modern automobile and demonstrate the fragility of the underlying system structure. They prove that, at a certain point in time, an attacker able to infiltrate virtually any ECU can leverage this ability to completely circumvent a broad array of safety-critical systems, and to control a wide range of automotive functions, including disabling the brakes, selectively braking individual wheels on demand, stopping the engine, and so on. The same authors, in [22], analyse the external attack surface of a connected car. In particular, they define the attack surface presented by the I/O channels of the car, i.e., indirect physical access (ODBII port, entertainment system), short range wireless access (Bluetooth, remote keyless entry, tire pressure, RFID car keys, and emerging short range channels such as hotspots for WiFi access), and long range wireless access

(broadcast channels, and addressable channels such as the remote telematics systems provided by manufacturers to offer a wide range of services such as crash reporting). The authors describe challenges in solving the vulnerabilities due to the listed attack vectors. Brooks et al. [18] provide a comprehensive survey of potential attacks on automotive software. The authors of [79] perform a survey of the intra-vehicle cyber-security threats from the point of view of the distinct protocol used in a car: Controller Area Network (CAN), Local Interconnect Network (LIN), FlexRay, and Media Oriented System Transport (MOST). More recently, a historical perspective and review of intra-vehicle communication networks has been provided in [63]. The attack vectors considered by this paper are similar to the ones defined in [22], with the addition of the USB port that could be installed in the car, the car to car communications, the web browsing, and the installation of application on the car entertainment system.

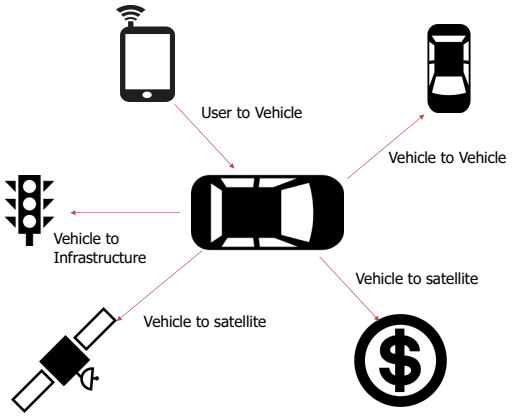

Fig. 1 Examples of vehicle communications.

Hereafter, we survey in detail the different security threats in the automotive domain, grouped with respect to the entities involved in the communication and the exploited communication technologies. Several communication technologies and means, as shown in Fig. 1, have been designed and developed [64] in order to establish communication intra and inter vehicles. Vehicles usually embed a large set (around 100) of ECUs communicating among themselves for operating the vehicle. Moreover, modern vehicles are able to communicate with other devices through wired or wireless interfaces such as USB, Bluetooth, WiFi, etc. These interfaces may expose the internal network to the outside world and can be considered entry points for cyber-attacks, which could impact the vehicle security and safety as well. Hence we proceed by discussing the following four automotive communication domains: 1) Ve-

hicle to vehicle (V2V), 2) Vehicle to infrastructure (V2X), 3) User to Vehicle (U2V), and 4) Intra Vehicle (Intra-V).

3.1 Vehicle to Vehicle (V2V)

In [22], the authors provide a possible classification of cyber security attacks in the automotive domain. *Remote attacks* are the ones that deal with vehicle-to-vehicle communications, such as eavesdropping on the communications, the forward of fake data to a vehicle to trigger an inappropriate reaction and, finally, damage of the ECU responsible for vehicle-to-vehicle communications, as shown in Fig. 2. This kind of attacks is very important and the need to cope with it stimulated the establishment of the "Car2Car Communication Consortium[5]" by six European car manufacturers. The goal of the consortium is to create a European industrial standard for car-to-car communications to be adopted across all brands. Communication between two vehicles

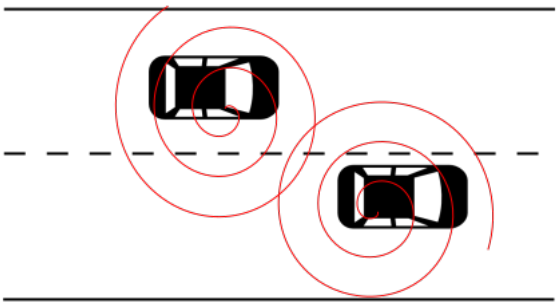

Fig. 2 Graphical representation of a V2V communication.

mainly happens through Ad Hoc networks, named Vehicular Ad Hoc Networks (VANETs). Referring to [72], the most important objective of a VANET is to support communications between different vehicles to improve the driver experience and make driving safer. Many protection mechanisms and frameworks have been developed to enforce security properties in VANETs. A classification of security and privacy issues related to VANETs is provided in [72]. The paper also presents several security solutions able to cope with different issues, such as generic security mechanisms that work for any mobile ad hoc network. Some specific solutions for the automotive domain have been also developed. For instance, European projects, such as SeVeCOM [55,83,68], PRESERVE[6], and EVITA[7] aimed at designing secure communication architectures for intra or inter-vehicular communications. In particular, the SeVeCOM EU project [55,83] developed a modular system supporting security and privacy features

[5] www.car-2-car.org
[6] https://www.preserve-project.eu/
[7] http://www.evita-project.org/

in V2V Ad Hoc Networks. In this solution, the network stack integrates a component to generate and verify signatures based on Public Key Infrastructure (PKI) with short term key and certificates in order to guarantee the authenticity of communication partners. Specifically, Inter Layer Proxies have been inserted into the network stack to intercept communication messages. In contrast, the authors of [59] propose a protocol for securing communication in VANETs, which exploits certificateless signature. This protocol exploits a reputation server, which is responsible for the distribution and management of identities and cryptographic credentials of vehicles. Further approaches, in [23, 54], do not require authentication to guarantee the authenticity of a broadcast announcement. These approaches take into account the number of received messages with the same warning; if this number is greater than a threshold, the announcement is considered true. The authors of [56] present a solution to protect VANET communications from roadside attackers, i.e., attackers who are located on the road with a computer to send fake messages to the cars passing on this road. The proposed approach is focused on the cooperative position verification to support cooperative safety applications. The work in [51] evaluates the performance of VANET when applying Private Key Encryption, but does not describe how to integrate the encryption support within the car communication system.

The work in [79] gives also an overview of the protection mechanisms that could be adopted as countermeasures to the threats and attacks the authors defined. A report, [41], has been recently published by the Institution of Engineering and Technology (IET), based on inputs from the Automotive Cyber Security Thought Leadership event (held in November 2014). It identifies the issues of automotive cyber-security and provides some recommendations on how to address them to ensure that future connected vehicles remain safe and are more efficient.

3.2 Vehicle to Infrastructure (V2X)

Requirements for information security in a vehicle are growing along with the complexity of vehicle functions. In addition to protecting internal vehicle data, vehicle connections to the outside world, as in V2X infrastructures (see Fig. 3) require heightened protection against unauthorized access. An example of such protection is *Intelligent charging*[8], that allows secure communication with an electric charging station. This solution is able to guarantee an authenticated data transmission among vehicles and infrastructure, Internet access and hotspot for infotainment in the vehicle, and so on.

Vehicle to roadside communication (V2R) is a particular case of V2X that also includes communication between a vehicle and roadside, service, home, and others. The main goal is to share information in order to enhance the safety of mobility to improve traffic efficiency, and to satisfy the need for convenience. V2X communication can be done through VANET.

[8] http://vector.com/vi_security_solutions_en.html

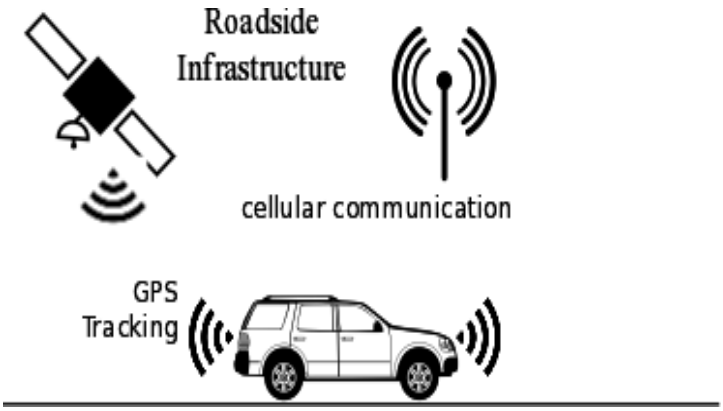

Fig. 3 Graphical representation of a V2X communication.

In [4], the U.S. Department of Transportation identified several application scenarios where VANETs can be useful. These applications can be categorized in safety-critical applications and non safety-critical ones. Basically, those safety-critical are related to traffic, congestion and so on. This points out how securing VANETs is important. For this reason in [72], the authors provide a list of all possible adversaries in any security system. It is worth noticing that security issues of the V2X communication are mainly related to the WiFi connection. Indeed, in [24] the authors state that, even though V2V and V2X are both wireless mobile networks, they are different and participate to the vehicular network with different communication technologies.

3.3 User to Vehicle (U2V)

Even though users can be considered as an external system with respect to a vehicle, communication User to Vehicle (U2V), as shown in Fig. 4, can be also treated as a particular case of V2X communication. Indeed, the work in [22] classifies as *remote attacks* also the ones that use short-range wireless communication technologies. Such attacks can be classified as direct, if they target a car's communication module, or indirect, if they target a driver's device that is connected to the car (e.g., a smartphone). Even though it is possible to use different technologies, such as Bluetooth and Dedicated Short Range Communications (DSRC), attacks may be: i) malicious or rational; ii) passive or active, such as eavesdropping (passive) or modification of data stream in order to create fake stream of data (active); iii) local or extended to more than one vehicle in such a way to violate, for instance, the privacy of

data. Several examples of remote attacks in the automotive domain have been

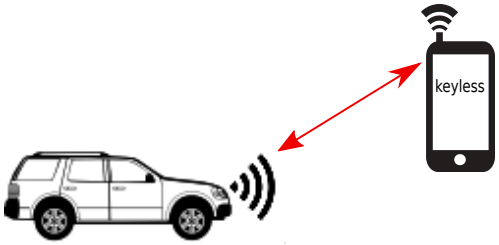

Fig. 4 An example of U2V communication.

surveyed in [62]. Among the others, the *Remote Keyless Entry/Start* system is based on key fobs of the user that contain a short-range radio transmitter that communicates with an ECU in the vehicle by sending encrypted data. Such data contain identifying information from which the ECU can determine if the key is valid and subsequently lock, unlock, and start the vehicle. A typical attack to this kind of system is the Denial of Service attack (DoS) that would not allow the car to be remotely locked/unlocked/started and in some cases it may be possible to unlock/start the car without the proper key fob. A real world example of DoS attack is the one to the Toyota Prius, described in [62]. Indeed, the DoS attack, also according to [72], is one of the most common attacks in VANETs and affects the availability of the system by bringing down the VANET, thus potentially causing an accident.

3.4 Intra Vehicle (Intra-V)

On the opposite side with respect to V2X communications, there are the Intra Vehicle communications, as shown in Fig. 5, that are related to the communications among distinct ECUs of the same vehicle. Intra-V communication technologies can be classified in four major groups [64]: 1) current wired, 2) multimedia, 3) upcoming wired, and 4) wireless. According to [63], the use of networks for communications among ECUs of a vehicle in production cars dates from the beginning of the 90s. Different suppliers have different requirements. This leads to the development of a large number of automotive networks as LIN, CAN, CAN FD, FlexRay, MOST, and so on. Indeed, an automotive system consists of several subsystems, each of them composed by around 100

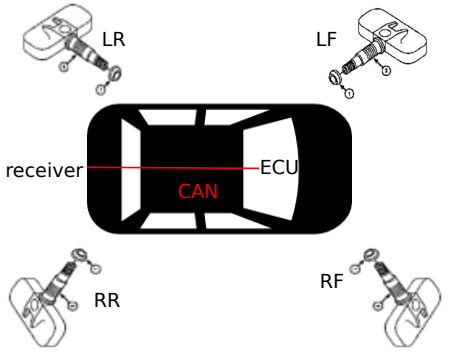

Fig. 5 An example of Intra-V communication (monitoring of tires pressure).

ECUs that communicate with one another. This makes the automotive system complex in many ways, including networking. To deal with this issue, the automotive industry recently set up several large consortia in order to agree on a common scalable electric/electronic architecture, such as AUTOSAR [12] and other standards we discuss in Section 4.5, and a common scalable communication system, such as FlexRay.

Different communication technologies are subject to different security issues. In [22], a classification of possible cyber-security attacks perpetrated by using internal automotive networks is provided. These attacks are classified as internal or local. *Internal attacks* exploit the vulnerabilities of the current network protocols, with a particular focus on CAN that was shown not able to guarantee confidentiality, authenticity, availability, integrity and non-repudiation security properties. *Local attacks*, instead, are performed by directly sending packets on an embedded bus (usually a CAN bus). This can be done by plugging an additional device on the targeted bus or through the On Board Diagnostic (OBD) port, that can be seen by an attacker as a plug-in entry point into the CAN bus. Examples of internal vehicle communications are communication with tire monitoring systems (e.g., via Bluetooth), transmission of authenticated messages (e.g., by Secure On-board Communication, SecOC for short, to prevent manipulation of critical signals) and so on.

4 Secure Software Engineering Process in Automotive Systems

As depicted in the previous sections, in automotive systems, a variety of applications work together or in parallel, encompassing multimedia and telematic ones, those specifically conceived for security and safety, and those concerning infrastructure and real-time event-based software management [70]. Additionally, the widespread usage of ICT facilities represents a new important dimension in the complexity of automotive systems, making the overall design and implementation difficult and critical. Addressing safety and security aspects is therefore becoming a stringent need in the automotive environment

and finding solutions, which may support verification of safety and security interactions in early stages of automotive development, is a real and urgent issue to be solved.

In such a situation, leveraging from software engineering best practices, the successful development of high quality systems characterized by complex interactions and safety needs calls for the application of a systematic and well organized software development process. In this vision, the objective is not just designing the system, but it is a more complex engineering discipline focusing on the production of maintainable, reliable, efficient, secure, and cost effective software. Considering the complexity and criticality of the automotive systems, the adoption of software engineering techniques and technologies could represent a key solution for a secure software development.

Over the last years, several studies have been carried out for investigating the specific challenges of an automotive software development process, such as the pillar analysis of [19] or the most recent works of [75,71]. As highlighted in these studies, improvements are necessary at several levels and phases of the process development.

In particular, the application of model based methodologies for guaranteeing the implementation of security requirements is becoming the successful key point for realizing customizable, extensible, and adaptable supports (like, for instance, a tool chain) for the application of *security-by-design* through the different stages of the development process of the automotive systems.

Without referring to any specific software development process, adopted by the different automotive enterprise or software industries, we report here below a brief description of the basic phases that can be identified as in [77]:

- **Requirements Analysis**: Services, constraints, goals, and features of the overall system are established and organized. The list of requirements specifies the guidelines that the project must adhere to. At the end of this phase a set of documented, actionable, measurable, testable, and traceable requirements should be defined to a level of detail sufficient for system design.
- **System and Software Design**: This phase targets both hardware and software requirements and establishes the overall system architecture. The representation of the system is usually provided in terms of functionalities easily transformable into one or more executable applications/programs. Depending on the target of the project, system design could include sketches, flow charts, site trees, HTML screen designs, prototypes, photo impressions, and UML schemas.
- **Implementation**: The established design is implemented in order to satisfy the list of requirements identified in the requirement analysis and definition phase. The implemented units, or subsystems are integrated into a complete system to be properly tested.
- **Validation and Testing**: Quality attributes as well as requirement assessment are calculated and validated. Different test techniques can be

included in the process development, each one targeting a different aspect of the proposed system.

In the rest of this section, for each of the above mentioned software development phases, the main approaches and tools specifically addressing security issues in the automotive sector are presented, and research challenges are outlined. Finally, the main standards and tool chains currently supporting the secure automotive software development are presented.

4.1 Requirements Analysis

Due to the growing innovative car functions, requirements engineering is one of the crucial issues in the automotive domain. Requirements are often described in specification documents containing an informal textual description [76]. The ECUs are typically developed by different suppliers and, to guarantee their integration and correct interpretation, a requirements engineering process producing detailed specification documents is needed. Many of the existing works report experiences and empirical studies about requirements management in the automotive industry [6,38,37]. The main factors influencing requirements analysis in the automotive domain are: i) late requirements and requirements changes, i.e., the requirements analysis could take several years and many requirements can change due to the long development process of a new car model; ii) requirements reuse, because available model independent function specifications are re-used for initial specification versions and are then augmented with model specific requirements; iii) incremental development; this implies that the overall vehicle development is incremental according to defined vehicle prototype levels with certain quality criteria; iv) the existence of many stakeholders involved in the realization of a car; this requires negotiation meeting and detailed documentation of the results. In addition, the authors of [38] identify three main generic phases of the requirements engineering process for the ECUs development: i) internal product targeting in which the scope, required functions, and key requirements of the ECUs are defined; ii) creation of a call-for-tender specification in which, based on the defined product scope, a dedicated specification document is created, mentioning all cost sources influencing the ECU cost; iii) bidding and contracting in which the call-for-tender specification is first distributed to potential suppliers for evaluation, and after the requirements negotiation, supplier selection, and signing of the contract are pursued.

The most used language for the requirements specification in the automotive domain is Electronic Architecture & Software Tools Architecture Description Language (EAST-ADL2) [27], which is defined for vehicle embedded electronic systems development and provides a unified notation for all the actors of a car development (car-maker, suppliers, etc.). In particular, EAST-ADL2 allows the decomposition and the modelling of an electronic system through five abstraction levels (Feature, Analysis, Design, Implementation, and Operational). It offers direct language support for the specification of safety re-

quirements and their allocation, the definition of system functions together with their behaviours and environmental assumptions, and the description of hazardous events, system malfunctions, and fault models. Moreover, EAST-ADL2 complements the AUTOSAR standard [12] with additional levels of abstractions and concepts, such as requirements engineering and safety, and it allows for annotating AUTOSAR elements with requirements and error models. Model-based approaches rely on EAST-ADL2 language for requirements specification. Specifically, the authors of [5] propose a methodology that directly includes the requirements in models and uses model driven engineering for a twofold objective: i) enriching the requirement expression to take into account multiform and multi-user requirements; ii) reducing the gap between the specifications and a solution model, by enriched traceability mechanisms. This methodology leverages the advantages of two UML2 profiles: Modelling and Analysis of Real-Time Embedded systems (MARTE) [66] mainly for timing properties expression, and System Modelling Language (SysML) [65] that defines how to express requirements with specific requirement diagrams and how to deal with traceability concerns defining relations between requirements, as well as requirements and model elements. Validation techniques are finally applied to these models to express the satisfaction of the requirements by the proposed solution. Other successful languages used for the specification of requirements in the automotive domain are: *Controlled Natural Language* (CNL) and its extended version, called *requirements patterns*, that allows to describe a subsystem and function hierarchy including required inputs and provided outputs for the subsystems and functions.

As highlighted in Section 2, with the advent of new wireless communication interfaces, the automotive networks are exposed to attacks originating both from the outside and inside the vehicle. This requires a careful analysis of security requirements in order to prevent system attacks and put in place cost-effective security measures. The work in [37] presents a security requirements analysis process applied to automotive on-board networks with V2X communication interfaces within the European research project EVITA [32]. It involves three main phases: i) identification of threats; ii) identification of security requirements including availability, timing and dependencies between requirements, by an abstract functional path approach at an early development process stage; iii) assessment of the risk level associated with the potential attacks and prioritization of the identified security requirements based on the results of the risk assessment. The TTool, presented in [37] is an opensource UML toolkit using several UML profiles and supporting the proposed methodology.

Other tools for requirements analysis in the automotive domain that are more focused on safety analysis are CHESS [7] and Medini Analyze [42]. The former is an Eclipse based framework that supports modelling and analysis across all development phases: from the definition of requirements, to the modelling of the system architecture, down to the software design and its deployment to hardware components, including tuning on the model in order to satisfy real time and dependability requirements. The latter is an

Table 1 Contributions on Requirement analysis.

	Projects				**Tools**		**Academic Research**			
	Sesamo[8]	AUTOSAR[12]	EAST-ADL2[27]	EVITA[32]	Medini Analyze [42]	CHESS[7]	[5]	[6]	[37]	[38]
Formal Representation	**X**		**X**	**X**	**X**	**X**	**X**		**X**	
Traceability Analysis		**X**	**X**		**X**		**X**			**X**
Reqs Verification	**X**	**X**	**X**		**X**	**X**	**X**			**X**
Reqs Integration		**X**	**X**					**X**		
Change Management	**X**	**X**				**X**		**X**		**X**
Reqs Analysis	**X**			**X**				**X**	**X**	**X**
Reqs Reuse		**X**			**X**					**X**
Security Issues	**X**			**X**		**X**			**X**	
Risk Analysis				**X**	**X**			**X**	**X**	
Compliance with standards	**X**	**X**	**X**		**X**		**X**			
Tool support	**X**	**X**	**X**		**X**	**X**	**X**			
Multiuser management		**X**			**X**		**X**			**X**
Recommendations								**X**		**X**

analysis and design toolset for automotive functional safety, able to derive functional and technical requirements from safety goals. It is tailored to ISO 26262 [45], integrates system architecture design in SysML [65] and software functional design (MATLAB/Simulink/Stateflow). It has been extended to integrate threat analysis, attack trees, security failure mode and effects analysis, fault tree analysis, and failure rate prediction. A first attempt to integrate both previous tools in a tool chain has been provided by the SESAMO project [8], with the aim of allowing integrated analysis of safety and security in order to reduce interdependencies between safety and security mechanisms employed to ensure these properties. More details about these tools are in Section 4.5.

In Table 1, the contributions reported in this section are summarized. In particular, the different solutions have been categorized into three different groups (Projects, Tools and Academic Research) as reported in the columns, and for each contribution 13 different dimensions have been considered as reported in the rows.

4.2 System and Software Design

The main research direction for addressing security in system and software design is model driven security that specializes Model Driven Engineering (MDE) for supporting the development of secure systems [13]. In MDE, models abstract all concepts of interest and their relations within a specific application domain. In the automotive domain, despite the increasing vehicles vulnerabilities, the design of automotive architectures is still mainly driven by safety and cost issues rather than by security issues [74]. Moreover, different specialized models with different abstraction levels are used for specific aspects at different development stages. The work in [17] presents a first attempt of model based development process in the automotive domain aiming at integrating the automotive concepts represented in EAST-ADL2 architecture description language [27] and the safety analysis tool HiP-HOPS [1]. This last supports iterative safety and dependability analysis by automatic synthesis of Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA), combination of results and creation of local fault trees. However, the main goal of [17] is to allow early safety analysis and system redesign by using model transformation techniques and leveraging the advantages of translating relevant information from the automotive domain to the safety analysis domain.

A big challenge in model based design is the assurance of consistency and traceability between artifacts of different development phases and viewpoints. A seamless model-based development methodology aligned with Automotive SPICE [9] is presented in [39]. It uses systematic and partially automated transitions between different perspectives and extends the system architecture to include all the information for simulation of the software already in the early stages of the development process. The proposed approach also enables an automated requirements validation as well as translation of requirements information to an initial SysML [65] system architecture model to be further manually refined. This model is extended by Real Time Statecharts (RTSCs) on the basis of Timed Automata, in order to specify and verify reactive and ECU timing requirements and, finally, to transform them into AUTOSAR model [12]. All transformations of a model are based on bidirectional Triple Graph Grammars (TGGs) that allow models synchronization and round-trip capabilities. Finally, the authors of [58,78] aim at establishing a model-driven system and safety-engineering framework to support the seamless description of safety-critical systems, from requirements at the system level to final hardware and software implementation. The approach relies on the integration of special purpose software and hardware development tools with a model based toolchain focused on domain standard exchange formats (such as AUTOSAR). The main goal is to establish interfaces between the various special-purpose tools involved in the system development process and bridge the existing gap between system design and software implementation tools for multi-core systems, minimizing redundant manual information exchange among tools. The proposed model transformation framework automatically generates software architectures in Matlab/Simulink, described via high level control system mod-

Table 2 Contributions on System and Software Design.

	Projects	**Tools**	**Academic Research**				
	MEMCONS [58]	HiP-HOPS [1]	[13]	[17]	[39]	[74]	[78]
Model-based design	**X**		**X**	**X**	**X**	**X**	**X**
Compliance with Standards	**X**			**X**	**X**		**X**
Safety	**X**	**X**		**X**	**X**		
Security			**X**			**X**	

els in SysML format and supports a consistent and traceable refinement from the early concept phase to software implementation.

Concerning security aspects in automotive design, the authors of [74] outline the upcoming security challenges in automotive architecture design and discuss the application of a model-based design approach, in combination with formalized verification methods, aimed at checking and avoiding vulnerabilities already during the design process of future automotive architectures based on Ethernet/IP.

Table 2 summarizes the main features of the projects, tools and academic works described in this section.

4.3 Implementation

In the automotive domain, coding is a complex and time-consuming phase. Suppliers produce most of the code for automotive systems, whereas the OEM carries out code for some of the infrastructures (such as bus gateways or communication backbones) or for innovative applications (such as advanced driver assistance) [20].

Model-based code generation is extensively utilized throughout the automotive software engineering community for its ability to address complexity, productivity, and quality challenges, although a lot of code is still developed by hand. Examples of tools that are commonly used in automotive model-based development are: i) the Real-Time Embedded Coder from MathWorks [60] that generates C and C++ code for embedded processors and offers built-in support for the AUTOSAR software standard; ii) Advanced Simulation and Control Engineering Tool for Software Development (ASCET-SD) from ETAS [31] that supports model-based development of automotive software with real-time requirements and automatic generation of ECU production code; iii) TargetLink from dSPACE [29] that is a software system generating C code from the MATLAB/Simulink/Stateflow graphical development environment; and, finally iv)

Table 3 Contributions on Implementation.

	Tools						**Academic Research**				
	Simulink [61]	ETAS [31]	dSPACE [29]	ESTEREL [30]	MathWorks [60]	GENEAUTO [81]	[57]	[50]	[26]	[15]	[80]
Code automatically generated by models	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
C/C++ code generation			**X**	**X**	**X**	**X**		**X**	**X**		
Code optimization				**X**	**X**						
Compliance with standards			**X**	**X**	**X**						**X**
Verification of automatically generated code			**X**	**X**	**X**	**X**			**X**		

SCADE Drive by ESTEREL Technologies [30], which allows tailoring and optimization of code generation for the target processors.

There exist several works related both to automatic code generation for model-based languages in embedded systems and to the verification of automatic code generation [80,26,15]. These works, mainly focused on Simulink [61], address how to specify executable models that enable the design and refinement of conceptual models by simulating the behaviour of the functionality to be realized and how to derive a C, C++ implementation of the functionality to be realized using automated code generation.

However, automatic code generation in the automotive domain introduces many challenges for the tool vendors and many issues including efficiency, coding standards, readability of generated code, legacy code integration and supporting tools, need to be addressed to satisfy the production-ready software development requirements. Existing works that partially try to target these challenges are [57,50]. In [57], a code generation tool for embedded automotive systems, based on finite state machines, is presented. It allows to generate the ECUs code starting from an annotated finite state machine and to perform structural validation of the state machine as well as schedulability check to guarantee that all deadlines of the system are met. GENEAUTO [50,81] is a qualifiable automatic code generator transforming Simulink, Stateflow and Scicos models to MISRA C and Ada SPARK code for safety-critical systems. It integrates formal components and has been applied to real size automotive systems.

Table 3 summarizes tools and academic works described in this section classified according to five main dimensions.

4.4 Validation and Testing

The existing solutions for security assessment of software systems basically perform either static analysis (i.e., the software is not executed), or dynamic analysis (performed by executing the software on specific inputs), although any life cycle will likely apply a combination of both. A 2009 survey carried out within the NIST SAMATE project[9], provides an analysis of more than 70 tools either specifically conceived for security assessment or more generally for software correctness related to security, and catalogued them under the following categories:

- Static Analysis: aids analysts in locating security-related issues.
- Source Code Fault Injection: the source code is instrumented by inserting changes and then executed to observe the changes in state and behavior that emerge.
- Dynamic Analysis: refers generically to security testing approaches, such as coverage-based or profiling.
- Architectural Analysis: aims at identifying flaws in the software architecture and determining resulting risks to information assets.
- Pedigree Analysis: identifies software coming from an external source (e.g., open source software).
- Binary Code Analysis and Disassembler Analysis: both categories review binary code. Since source code is not needed, they can be applied to Commercial Off-The-Shelf (COTS) components.
- Binary Fault Injection: focuses on likely faults in the real-time operation of the software (e.g., memory faults or other error conditions provided by the processor).
- Fuzzing: a form of negative testing, in which the software is exercised under random invalid data, generally specific to a particular type of input.
- Malicious Code Detectors: search for malicious logic embedded in programs.
- Bytecode Analysis: bytecode contains more semantic information about the program execution than an equivalent binary. Bytecode analysis tools are actively investigated.

The study concludes that, at that time, the software security market was focused almost entirely on source code analysis, which is necessarily limited by the implementation language. Nowadays, more and more in industrial application the process of modelling and identifying vulnerabilities is embedded into the software development life cycle and is moving towards early security assessment or even to guaranteeing security-by-design. A first stage of risk assessment is used to identify high-risk areas or features, and thus determine and optimize the respective security assessment effort. Then, the testing phase is extended with risk-based security testing. Several tools for security test modelling, execution, and analysis have been released by the ITEA project DIAMONDS [49] and EU FP7 Project Rasen [34]. In model-based design and

[9] http://samate.nist.gov

Table 4 Contributions on Verification and Testing.

	Projects			**Tools**				**Academic Research**			
	DIAMONDS [49]	Rasen [34]	SeVeCom [83]	TTCN3 [2]	DIANA [43]	SPIN [11]	Simulink [61]	[21]	[25]	[52]	[73]
Design Verification	**X**						**X**		**X**		
Code Verification	**X**						**X**		**X**		
Fault injection						**X**					
Risk assessment analysis	**X**	**X**									
Architectural analysis			**X**								
Safety vs Security					**X**	**X**		**X**		**X**	**X**
Compliance with standards			**X**	**X**		**X**		**X**	**X**		**X**
Safety HW/SW					**X**	**X**					
Communication Security	**X**	**X**	**X**		**X**	**X**		**X**			
Testing	**X**	**X**		**X**	**X**						

production code generation, two main validation and testing activities can be distinguished: i) design verification at the model level to demonstrate that the model is well-formed, meets its requirements, and does not contain unintended functionality; ii) code verification using equivalence testing and other techniques to demonstrate equivalence between the model and the generated code compiled into an executable model.

Many works try to transfer existing security assessment methods to the automotive domain, addressing specific risks and challenges of automotive software. Nowadays, several works [52,21,73] concur that security cannot be addressed separately from safety in the software engineering process of automotive systems, thus assessment approaches should converge and complement each other.

The widespread utilization of model-based design and production code generation in the automotive development process enables the application of automatic verification and validation techniques earlier in the software lifecycle, and at a higher level of abstraction using, for instance, Simulink models [61]. Specifically, the work in [25] discusses a verification and validation workflow

of models and generated code for developing in-vehicle software components compliant with the objectives and constraints of ISO 26262-6 [45] using model-based design. It discusses tool support by using a Simulink family tool chain for model-based design and provides as instantiation of the verification and validation requirements outlined in ISO 26262-6 [45].

Many testing methodologies and tools are customized to address specific challenges of the automotive domain. Specifically, Testing and Test Control Notation (TTCN3) [2] is a language standardized by ISO for the specification of tests for real-time and communicating systems, developed within the framework of standardized conformance testing (ISO/IEC 9646) [47]. It has been used in the SeVeCom project [83] to describe the test architecture, stimulate the security communication stack with relevant events and check its behaviour. Specifically, SeVeCom developed a modular architecture for integrating into the network stack a system for generating and verifying signatures based on Public Key Infrastructure (PKI) with short term key and certificates in order to guarantee the authenticity of communication partners.

Among testing tools, Digital Instrument for Automatic Network Analysis (D.I.A.N.A) [43] is a solution to automatically validate the automotive ECU CAN network from the physical, communication, and diagnosis perspectives. Through D.I.A.N.A, an analysis of the traffic flow on the CAN bus can be done to evaluate the performance, to mark potentially risky messages, and to test the legitimacy of messages by understanding if data circulating on the bus are coherent and permitted in accordance with the operational state of the vehicle.

Other security testing activities are aimed at finding implementation errors that could be exploited by an outside attacker to cause potential functionality problems. Such activities are also used to establish to what extent the target system can resist an attack and generally consist of: i) functional automotive security testing able to ensure general compliance with the specifications and standards for the security functionality implemented, encryption algorithms and authentication protocols of a vehicular IT system; ii) vulnerability scanning applied to all relevant applications, source codes, networks, and back-end infrastructures of an automotive system in order to test the system for common security vulnerabilities such as security loopholes or security configurations with known weaknesses taken from a continuously updated database of automotive security vulnerabilities; iii) fuzzing that is used to expose the implementation to unexpected, invalid, or random input in the hope that the target will react in an unexpected way and, as a result, uncover new vulnerabilities; iv) penetration tests, applied in a final step to test security of the whole system by applying attacking methods conducted by trusted individuals to check whether ECUs are vulnerable and could allow unauthorized users access.

Specifically, a fault injection based validation methodology has been developed by Magneti Marelli [11] for assessing the compliance of the vehicle with the ISO 26262 standard. The methodology allows to identify the injection points as safety-critical data interfaces between both hardware (HW) and

software (SW) blocks and to perform code insertion at runtime into the target program. Typical inserted faults are the substitution of the value computed by ECU algorithms with a fixed one before sending it on CAN or the missing counter update preventing the publication of the ECU output on the CAN network. Moreover, this fault injection approach can be conducted on the vehicle without exposing the driver to harm and is effective in checking safety mechanisms against timing constraints.

In Table 4, we provide a graphical representation of a classification of existing solutions with respect to some of the criteria listed above plus some additional criteria coming from security and safety aspects of automotive systems.

4.5 Standards and tool-chain support for secure automotive software development

Due to the high cost and effort impact of ECU coding, integration, and testing, in the last years an intense activity of standardization and tool-chains definition has been performed. Indeed, the availability of dedicated standards and tools makes the assessment and reuse of code, as well as the definition of generic best practices, easier. In the automotive community, the most widespread standards addressing component design, implementation and process issues can be summarized as in the following:

- **AUTOSAR** [12] is an initiative born by an alliance of OEM manufacturers and Tier 1 automotive suppliers working together to develop and establish a de-facto open industry standard for automotive Electric/Electronic architecture, which will serve as a basic infrastructure for the management of functions within both future applications and standard software modules. The goal is to provide a common software infrastructure for automotive systems based on standardized interfaces for the different architecture layers. The resulting standard architectural approach aims at achieving the technical goals of modularity, scalability, transferability and re-usability of functions. AUTOSAR provides implementation and standardization of basic system functions including, e.g., bus technologies, operating systems, communication layer, HW abstraction layer, memory services, mode management, middleware/interfaces, standard library functions as well as integration of functional modules from multiple suppliers. AUTOSAR defines several safety and security features such as: i) memory partitioning; ii) protection against unauthorized use of basic software modules; iii) protection of safety related data against corruption using checksums; iv) end-to-end communication protection mechanisms; v) program monitoring to check the correct execution of software; vi) provision of synchronized time bases; vii) fault detection mechanism. Many extensions of AUTOSAR have been presented such as [14] that allows to specify reconfiguration aspects at the architectural level and to automatically derive the needed reconfiguration functionality based on the architectural information. These reconfiguration

capabilities make automotive systems more flexible and robust as well as able to deal with failures of sensors or attacks.

- **ISO26262** [45] is an international standard addressing functional safety features of each automotive product development phase, ranging from the specification, to design, implementation, integration, verification, validation, and production release. The standard ISO 26262 is an adaptation of the Functional Safety standard IEC 61508 for Automotive Electric/Electronic Systems [44]. As its parent standard IEC 61508, ISO 26262 is a risk based safety standard, where the risk of hazardous operational situations are qualitatively assessed and safety measures are defined to avoid or control systematic failures and to detect or control random hardware failures, or mitigate their effects. The ISO 26262 standard: i) provides an automotive safety lifecycle (management, development, production, operation, service, de-commissioning) and supports tailoring the necessary activities during these lifecycle phases; ii) covers functional safety aspects of the entire development process (including such activities as requirements specification, design, implementation, integration, verification, validation, and configuration); iii) provides an automotive-specific risk-based approach for determining risk classes (Automotive Safety Integrity Levels, ASILs); iv) uses ASILs for specifying the item's necessary safety requirements for achieving an acceptable residual risk; v) provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety is being achieved. The standard scope embraces the whole system life cycle and addresses specifically the hardware and the software development lifecycle. The introduction of such a new standard allowed to cover a relevant normative gap in the development of Electric, Electronic and Programmable Electronic (E-E-PE) systems in automotive. In fact, while in other domains (as for instance railways, avionics, etc.) functional safety standards have existed for decades, automotive functional safety has been insufficiently addressed and handled so far. In particular, addressing the software as a key player for the overall safety of E-E-PS systems is probably one of the most characterizing point of the standard.
- **Automotive SPICE** [10] is a standard widely used in the automotive domain, aimed at providing a framework for the evaluation of the capability of software development-related processes of producers of software-intensive automotive components. Such an approach is used to qualify suppliers of software-intensive automotive components on the basis of the technical adequacy, discipline, and technical maturity of their process.
- To face cyber-security challenges, existing related standards as the **ISO/IEC 15408** [48], and **ISO/IEC 27034** [46] need to be integrated with new ones as the IEEE 1609 family of standards for Wireless Access in Vehicular Environments (WAVE) [40]. Such a standard, which is going to be released, aims at defining an architecture and a complementary, standardized set of services and interfaces that collectively enable secure vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2X) wireless communications. Together these standards are designed to provide the foundation for a broad

range of applications in the transportation environment, including vehicle safety, automated tolling, enhanced navigation, traffic management, and many others.

As highlighted by these standards, there is a necessity of extending and integrating the adopted development process with procedures that impact jointly safety and security as well as the use of specialized supports, languages, and tools through the different phases of the software engineering process so to make easier the security-by-design adoption. To this purpose, recently several tool-chains have been proposed as innovative solutions for integrating different approaches, methodologies, and techniques related to security-by-design, model driven architecture, software process development in complex, critical systems. In the rest of this section, some of the main available tool-chains are briefly described. Although not explicitly tailored to the automotive domain, they could be conveniently customized and applied to the automotive area.

SESAMO tool-chain. SESAMO [8] is a cross domain tool-chain that provides support for implementation, integration, testing, verification and validation phases of the general SESAMO development process. The main tools are:

1. *CHESS* tool [7] is an Eclipse framework based on UML/SysML profile [65] that supports modelling and analysis across all development phases and provides also an editor facility. *CHESS* tool offers schedulability and dependability analysis functionalities: according to the analysis results that are back-propagated onto the model itself, the modeller can perform some tuning on the model to satisfy real time and dependability requirements.
2. *Medini analyze* [42] is an Eclipse based analysis and design toolset for automotive functional safety. It is specifically tailored to ISO 26262 and integrates system architecture design (SysML) and software functional design (MATLAB/Simulink/Stateflow). It integrates threat analysis, attack trees, security FMEA, fault tree analysis and failure rate prediction. *medini analyze* can trace and track all safety relevant information and decisions throughout the whole development process. It supports hazard analysis and determination of Automotive Safety Integrity Levels (ASIL), and derivation of functional and technical requirements from safety goals. System models defined and detailed in *medini analyze*, associated with safety/security goals and requirements, can be imported into the corresponding *CHESS* views.
3. The *Assurance and Safety Case Environment* (ASCE) [3] is a powerful flexible graphical and narrative hypertext tool used for the development, review and maintenance of assurance and safety cases and other structured technical documentation. It supports a variety of notations, including the widely-used Goal Structuring Notation (GSN) and Claims-Arguments Evidence (CAE) notation, as well as various argument modelling notations and causal analysis like fault trees and why-because analysis. Additional notations can be supported through the use of domain-specific schemas and

plug-ins can be used to import evidence from other tools and add support for graphical analysis techniques.

4. The *Preliminary Interdependency Analysis* (PIA) tool [69] provides a modelling environment and simulation engine for iterative system modelling and risk analysis. It specifically supports qualitative and quantitative analysis of interdependencies between components of complex systems such as critical infrastructures. Models can be analysed by probabilistic tools (system components are represented by continuous-time state machines), which allow the identification of different measures of interest such as the likelihood of cascade failures or the weakest link in the modelled system.

NESSoS tool-chain. Within the NESSoS project [33], whose main aim was the creation of a European common body of knowledge in the area of secure software development and service composition, a variety of analysis and development tools for different aspects of service security engineering has been provided and collected in [36]. These tools have been integrated in a common Service Development Environment (SDE) that is based on the Eclipse Integrated Development Environment (IDE) and enables a simple and lightweight integration of tools providing a common user interface and orchestration mechanisms. The SDE was designed to integrate tools that support all development phases in an engineering process including modelling, analysis, code generation, and runtime support. A goal of SDE is to encourage the use of automated model transformations from high-level models and formal specifications to model level, for supporting model-driven development of software and systems. An example of NESSoS tool-chain for secure software development is the Access Control Testing (ACT) tool chain described in [16]. ACT includes three main components: i) model-driven policy design for developing the graphical specification of access control requirements and converting it into an XACML policy; ii) test case generation and execution for deriving test cases and executing them on the XACML policy; iii) trace analysis and model compliance for analyzing the requests execution results and assessing their compliance with the access control model.

AMALTHEA4public [84] is an eclipse-based open source development platform for the development of hardware architectures like multi- and many-core platforms. Specifically, it focuses on the improvement of description capabilities of existing hardware and software models so to support the new system architectures and correctly manage the resulting level of parallelism and performance. The tool-chain integrates a variety of tools, each one focused either on a specific view of the system or on a particular step of development. The target is the automation of the data modifications and transition over the overall development process. Thus, through the AMALTHEA platform, the necessary information for the complete development process can be used and exchanged via well-defined interfaces between different tools. The data include both software and hardware descriptions, which are extended with elements for supporting multi-core aspects. To this purpose, a challenge of the tool-

chain is providing an open interface, such that different tools, whether open source, commercial or corporate-owned, can be integrated.

ARTOP [82] AUTOSAR TOol Platform is an infrastructure for the development of tools, which are used for the design and configuration of AUTOSAR systems. The core component of ARTOP is the AUTOSAR metamodel implementation supporting all available AUTOSAR meta-models. ARTOP includes the AUTOSAR XML schema conformant serialization, rule-based validation, model refactoring, workspace management, as well as example editors and further utilities that allow interoperability of different tools able to support the AUTOSAR methodology steps. ARTOP implements non-competitive base functionalities needed by each AUTOSAR tool and allows to develop commercial tools with better quality in less time, since only key functionalities have to be implemented. It has been successfully applied to an AUTOSAR configuration framework within Continental Engineering Services (CES).

Mx-Suite [28] is a comprehensive tool chain for verification and validation of embedded systems. It provides many facilities such as intuitive graphical diagrams for software requirements specification and documentation of performance constraints, as well as tools for simulation of a complete vehicle electrical system. It allows to automate tests, to perform regression and coverage-based testing, to define testable requirements as well as improve their traceability and detection of possible failure causes. Mx-Suite can be used by OEMs and Tier-1 suppliers to test simulation models, integrate ECU modules and test them by also importing ECU test and log data from other tools as test cases. In particular, Mx-Suite supports the testing of AUTOSAR software components allowing to integrate them in a software architecture and to run them in a Software In the Loop (SIL) test environment.

5 Research Challenges

The hacking of vehicular communication may lead to scenarios with serious or even catastrophic consequences for society. A plausible scenario could be the one in which an attacker contemporaneously stops all refrigerator trucks carrying pharmaceutical material, turning off their fridges and eluding the cargos remote monitoring system, with dramatic consequences for patients and citizens in vast areas. An even more critical situation could be the case in which terrorists exploit weaknesses and vulnerabilities of car systems for their malicious purposes, such as vehicle ramming attacks or vehicular assaults aiming at breaching buildings, or detonating explosives.

The above depicted scenarios highlight the need to radically revisit the overall development of secure and safe applications in the automotive sector. A “safe and secure by design” paradigm [8] is strongly promoted to enhance robustness of systems operating in hostile environments, where both accidental faults and intentional attacks are to be handled.

Table 5 summarizes the main projects, tools, tool-chains and academic works for automotive security concerning each phase of the software engi-

Table 5 Software engineering lifecycle phase v.s. automotive communications domains

	Requirements	**Design**	**Implementation**	**Verification&Testing**
V2V	[32], [8]	[8]	[8]	[83]
V2X	[37],[32]		[83]	[83], [49]
U2V	[8]	[8], [33]	[8], [33]	[33],[73],[21]
Intra-V		[84],[82]	[84],[82]	[43], [73],[11], [25], [28]

neering lifecycle with respect to the four automotive communication domains defined in Section 3. Based on our review of existing solutions, a gap exists between automotive cyber security requirements and the current software engineering solutions tailored for the automotive domain. For instance, for what concerns the System and Software design phase, the approaches presented in Section 4.2 are not tailored for any specific automotive communication domain. However, they are generic enough to be considered applicable to all the four domains.

In the following, the major identified challenges are discussed.

Safety and Security convergence. As said, fostering the convergence of safety and security is one of the main challenges of nowadays automotive systems [21]. Moving beyond the basic vision of integrated safety and security modelling solutions, a holistic approach in which safety and security interactions are addressed early in the different stages of the automotive development is needed [35]. Problems and issues may arise from the different points of view between safety and security. Usually functional safety targets mainly systematic and random hardware failures as possible source of security threats, while security approaches focus on attacks that cause unauthorized access and manipulations. For instance, vehicle safety assumes that for the sake of a quick analysis and periodic controls, systems resources, such as Random-access memory (RAM), should be easily accessed, whereas security would restrict such an access, as much as possible, via authentication and authorization mechanisms. Thus adopted security solutions should be carefully integrated and compliant with safety and vice versa.

Tool-chain development. For improving efficiency, it would be desirable to integrate open-source tools for systems engineering, verification, and test generation. However, the inherent complexity of employed software systems, which also adopt distributed, heterogeneous pieces of software functionalities, inhibits the complete automation of automotive software development. Indeed, current automated solutions focus only on small part of the software development process or are not specifically tailored to the automotive environment. Thus, an important challenge is the possibility to integrate into a complete tool-chain all the activities related to the development of automotive many-core systems.

Standardization. The adopted software engineering process should be extended and integrated according to the best practice standards, such as

ISO26262 and SPICE guidelines, and to the guidelines proposed by AUTOSAR. Gaps in the existing safety standards could be overcome by revising their definitions in the direction of making them valid also for security purposes. For instance, classes of severity and controllability defined in the safety standards could be modified so to include both safety and security.

Communication technologies integration. According to [64], it is desirable that the set of networking technologies typically found in an automotive system will be interconnected. Such interconnection would provide timeliness, composability, and fault tolerance across the whole vehicle "network of network". The ECUs for controlling the vehicle functions communicate with other devices through wired or wireless interfaces such as USB, Bluetooth, WiFi, etc. On the other hand, these interfaces may expose the internal network to the outside world and can be considered entry points for cyber-attacks, which could impact vehicle physical security and safety. For instance, as already noticed, the vehicle owner may change car parameters, activate features, use patches or other software applications that may have an impact on privacy and security in general.

Secure Engineering process. Promoting an ad-hoc engineering process targeting jointly safety and security issues is another major challenge. In particular, targeting the enhancement of a software and system engineering process in order to include specific safety and security aspects will be an innovative research direction. For this, improvements are necessary on several levels and phases of the process development: a) the requirements specification and analysis should be better performed and properly supported by models/tools; b) the system and architectural design modelling should be detailed better so that it can be easily translated into code; c) the implementation should be continuously revised to reflect the many changes/updates to the design models; d) the verification and validation support should be completely automated, timely and cost effective. More in general, the procedures, activities, and methodologies currently adopted in the different stages of automotive software development need to be enhanced to enable the correct and complete definition of a secure-by-design architecture as well as the set of services and interfaces that collectively enable secure intra vehicle, vehicle-to-vehicle and vehicle-to-infrastructure communications.

In the following sections, we present a more detailed description and discussion of different challenges for each specific phase of the software development process together with a proposal of a secure-by-design toolchain.

5.1 Process Development Phases: Challenges for Connected Vehicles

In this section, the challenges that characterize each phase of the software development process in the automotive systems are tackled.

Requirements. Communication between OEMs and suppliers has to be centered on formal requirements documents, which should be sufficiently pre-

cise and complete. Moreover, it is desirable to find an integrated approach able to overcome all the key issues listed in Section 4.1, such as reuse requirements, incremental development, and traceability of requirements.

Design. As depicted in Section 4.2, the main issues of model-based design in automotive domain are models redundancy, inconsistency, lack of automation and the usage of specialized models with varying abstraction levels for specific aspects at different development stages. The automotive industry has a growing demand of integrating modelling notations and tools in a model based development tool-chain for embedded systems. A challenge is the assurance of consistency and traceability between artifacts of different development phases. To this purpose, an integrated and holistic models chain is advised, able to allow automated traceability between different models elements and requirements, as well as translation from textual requirements to model-based requirements and from the system architecture to the software architecture. This integrated models chain will cover several needs at design level phase, including: i) assure models consistency; ii) provide the facilities to manage large amount of requirements; iii) manage growing size of the models: and iv) give guidance about which models have to be used at a specific development stage. Another important challenge is the integration of security issues in the model-based automotive system design.

Implementation. Among the most important issues and challenges regarding model-based design and production code generation are the handling of large scale models (e.g., for autoscaling and testing) as well as the proper integration of those techniques in the whole development process for entire workgroups. Furthermore, generated code must be easy to integrate with legacy code and other application software in existing ECUs, and must conform to specified formats to facilitate readability and testability.

Validation. In model-based and risk-based security assessment, existing solutions are meant to generally test the robustness of the system against potential attacks and do not directly prevent any kind of automotive security attack. Holistic solutions are needed that can forecast and inhibit potential attacks based on model-driven testing approaches. Another challenge is to define a comprehensive framework that can address all different steps of the security validation process, from threat identification to security test generation and to architectural security assessment, and can also handle safety interactions.

5.2 Discussion on Research Challenges: Security-by-Design Toolchain Proposal

In this section, an example of a Secure-by-Design (SbD) tool-chain is provided, which relies on the results of the SESAMO project. The proposed approach has been promoted through standardization activities in other domains, especially avionics (RTCA/EUROCAE ED-202). As an extension of the SESAMO ap-

proach, the best practices in the software engineering research area have been exploited and customized in such a way that safety and security aspects can be automatically integrated into the architectural design. In particular the most innovative and technologically advanced correct-by-design approaches and methodologies have been included in the *SbD* design pattern in order to assure system-level safety and security properties.

However the demonstration of correctness through several techniques, such as deterministic property conditional on specific assumptions, probabilistic arguments demonstrating acceptable effects of violations of specific assumptions (e.g., by component failures) and additional arguments (empirical and logical) to support sufficient confidence in the remaining assumptions of both deterministic and probabilistic arguments, need to be further investigated. From the security perspective, the proposed *SbD* is able to develop specific and customized solutions in order to confront cyber-security attacks, detect anomalies in the automotive system and recover from them. Among the proposed solutions, continuous authentication and authorization mechanisms, as well as powerful protection mechanisms to enforce integrated safety and security requirements at design time are put in place. In this way, *SbD* aims to guarantee security in the automotive communication channels, in such a way that data confidentiality and integrity will be preserved. Specifically, *SbD* includes:

- Model-based methodologies and tools based on the most innovative engineering best practices, supporting the overall systems and software development process from requirements analysis to coding, verification, and assessment phases;
- The most effective tools and techniques for the conceptualization of automotive needs and the certification and validation of the marketplace outcomes;
- A strong validation and testing activity to certify and assess all the products and results obtained during the automotive software development.

Finally, feedback received from the operational testing and evaluation performed during the demonstration activities of *SbD* application to real case studies can promote suggestions for standardization bodies. These suggestions will focus on overcoming challenges about safety and security aspects in the automotive area.

6 Conclusion

In this work, we have discussed the impact of pervasiveness of ICT in automotive systems. Starting from the '70s, the embedding of communication technologies into vehicles is growing over and over. Even though researchers, automotive industries, and suppliers have been working together to provide innovative, safe, and as secure as possible solutions, several gaps between the

existing technological solutions and the automotive domain specific requirements still exist.

This paper reviewed existing techniques and tools addressing security issues in the different phases of the software development process pointing out their limitations to support a security-by-design paradigm for automotive systems. We identified open research challenges and hinted at desirable directions for future research in the area, aiming to overcome the identified problems.

Following this survey, and based on the results of the SESAMO project, we envision a secure-by-design adaptable and extensible software tool-chain that can support the development of enhanced security methodologies and techniques in all phases of the software development process, promoting an ad-hoc engineering process able to jointly target safety and security. This tool-chain would be a first attempt to overcome the identified open challenges in the automotive software development process in a way compliant with existing standards.

7 Acknowledgments

This work has been partially supported by the GAUSS national research project (MIUR, PRIN 2015, Contract 2015KWREMX).

References

1. HiP-HOPS - Hierarchically Performed Hazard Origin and Propagation Studies. http://hip-hops.eu/. Online; accessed May 2016
2. TTCN-3 - TESTING AND TEST CONTROL NOTATION VERSION 3. http://www.ttcn-3.org/. Online; accessed September 2016
3. Adelard LLP: ASCE Assurance & Safety Case Environment. http://www.adelard.com/asce/. Online; accessed May 2016
4. Administration, N.H.T.S., et al.: Vehicle safety communications project task 3 final report: Identify intelligent vehicle safety applications enabled by dsrc. DOT HS S09 S **59** (2005)
5. Albinet, A., Begoc, S., Boulanger, J., Casse, O., Dal, I., Dubois, H., Lakhal, F., Louar, D., Peraldi-Frati, M., Sorel, Y., et al.: The memvatex methodology: from requirements to models in automotive application design. In: 4th European Congress ERTS (Embedded Real Time Software), Toulouse, France (2008)
6. Almefelt, L., Berglund, F., Nilsson, P., Malmqvist, J.: Requirements management in practice: findings from an empirical study in the automotive industry. Research in engineering design **17**(3), 113–134 (2006)
7. ARTEMIS: JU CHESS Project. http://www.chess-project.org. Online; accessed May 2016
8. ARTEMIS: SESAMO - Security and Safety Modelling. http://sesamo-project.eu/. Online; accessed May 2016
9. Automotive, S.: Automotive spice, process assessment model (pam) v2.5. In: The Procurement Forum. The SPICE User Group (2010)
10. Automotive, S.: Automotive spice, process reference model (prm) v4.5. In: The Procurement Forum. The SPICE User Group (2010)
11. AUTOMOTIVE SPIN: Thirteenth Automotive SPIN Italia Workshop. http://www.automotive-spin.it/download.php. Online; accessed May 2016
12. AUTOSAR - AUTomotive Open System ARchitecture: http://www.autosar.org/. Online; accessed May 2016

13. Basin, D., Clavel, M., Egea, M.: A decade of model-driven security. In: Proceedings of the 16th ACM symposium on Access control models and technologies, pp. 1–10. ACM (2011)
14. Becker, B., Giese, H., Neumann, S., Schenck, M., Treffer, A.: Model-based extension of autosar for architectural online reconfiguration. In: Proceedings of International Conference on Model Driven Engineering Languages and Systems, pp. 83–97. Springer (2010)
15. Berry, G., Bouali, A., Fornari, X., Ledinot, E., Nassor, E., de Simone, R.: Esterel: a formal method applied to avionic software development. Science of Computer Programming **36**(1), 5 – 25 (2000)
16. Bertolino, A., Busch, M., Daoudagh, S., Lonetti, F., Marchetti, E.: A toolchain for designing and testing access control policies. In: Heisel et al. [36], pp. 266–286
17. Biehl, M., DeJiu, C., Törngren, M.: Integrating safety analysis into the model-based development toolchain of automotive embedded systems. SIGPLAN Not. **45**(4), 125–132 (2010)
18. Brooks, R., Sander, S., Deng, J., Taiber, J.: Automobile security concerns. Vehicular Technology Magazine, IEEE **4**(2), 52–64 (2009)
19. Broy, M.: Challenges in automotive software engineering. In: Proceedings of the 28th international conference on Software engineering, pp. 33–42. ACM (2006)
20. Broy, M., Kruger, I., Pretschner, A., Salzmann, C.: Engineering automotive software. Proceedings of the IEEE **95**(2), 356–373 (2007)
21. Burton, S., Likkei, J., Vembar, P., Wolf, M.: Automotive functional safety= safety+ security. In: Proceedings of the First International Conference on Security of Internet of Things, pp. 150–159. ACM (2012)
22. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T., et al.: Comprehensive experimental analyses of automotive attack surfaces. In: USENIX Security Symposium. San Francisco (2011)
23. Chen, L., Ng, S.L., Wang, G.: Threshold anonymous announcement in vanets. Selected Areas in Communications, IEEE Journal on **29**(3), 605–615 (2011)
24. Chou, C.M., Li, C.Y., Chien, W.M., Lan, K.c.: A feasibility study on vehicle-to-infrastructure communication: Wifi vs. wimax. In: Proceedings of Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, pp. 397–398. IEEE (2009)
25. Conrad, M.: Verification and validation according to ISO 26262: A workflow to facilitate the development of high-integrity software. Embedded Real Time Software and Systems (2012)
26. Conrad, M., Mosterman, P.J.: Model-Based Design Using Simulink Modeling, Code Generation, Verification, and Validation, pp. 159–181. John Wiley & Sons, Inc. (2013)
27. Cuenot, P., Frey, P., Johansson, R., Lönn, H., Papadopoulos, Y., Reiser, M.O., Sandberg, A., Servat, D., Kolagari, R.T., Törngren, M., et al.: 11 the east-adl architecture description language for automotive embedded software. In: Model-Based Engineering of Embedded Real-Time Systems, pp. 297–307. Springer (2010)
28. Danlaw Inc.: Mx-Suite. http://goo.gl/UJcMpX. Online; accessed May 2016
29. dSPACE: TargetLink Code Generator. https://goo.gl/sSG0mD. Online; accessed May 2016
30. ESTEREL: http://www.esterel-technologies.com/products/. Online; accessed May 2016
31. ETAS: http://www.etas.com/en/index.php. Online; accessed May 2016
32. EVITA: E-safety vehicle intrusion protected applications. http://www.evita-project.org/. Online; accessed May 2016
33. FP-7: NESSoS - Network of Excellence on Engineering Secure Future Internet Software Services and Systems. http://www.nessos-project.eu. Online; accessed May 2016
34. FP-7: Rasen - Compositional Risk Assessment and Security Testing of Networked Systems . http://www.rasenproject.eu/. Online; accessed September 2016
35. Glas, B., Gebauer, C., Hänger, J., Heyl, A., Klarmann, J., Kriso, S., Vembar, P., Wörz, P.: Automotive safety and security integration challenges. In: Automotive - Safety & Security (2014)

36. Heisel, M., Joosen, W., Lopez, J., Martinelli, F. (eds.): Engineering Secure Future Internet Services and Systems - Current Research, *Lecture Notes in Computer Science*, vol. 8431. Springer (2014)
37. Henniger, O., Apvrille, L., Fuchs, A., Roudier, Y., Ruddle, A., Weyl, B.: Security requirements for automotive on-board networks. In: Proceedings of the 9th International Conference on Intelligent Transport System Telecommunications (ITST) (2009)
38. Heumesser, N., Houdek, F.: Experiences in managing an automotive requirements engineering process. In: Proceedings of 12th IEEE International Conference on Requirements Engineering, pp. 322–327 (2004)
39. Holtmann, J., Meyer, J., Meyer, M.: A seamless model-based development process for automotive systems. In: Proceedings of Software Engineering (Workshops), pp. 79–88 (2011)
40. IEEE: IEEE 1609 - Family of Standards for Wireless Access in Vehicular Environments (WAVE)
41. IET, The Institution of Engineering and Technology: Automotive Cyber Security: An IET/KTN Thought Leadership Review of risk perspective for connected vehicles. http://goo.gl/2mhmvk. Online; accessed May 2016
42. ikv++ Technologies: medini analyze. http://goo.gl/kVPlp5. Online; accessed May 2016
43. Intecs SpA: D.I.A.N.A. http:www.intecs.itengprodotti_dettagli.asp?ID_Prodotto=30. Online; accessed May 2016
44. International Electrotechnical Commission: Functional Safety and IEC 61508. http://www.iec.ch/functionalsafety/. Online; accessed May 2016
45. ISO: ISO 26262 - Road Vehicles - Functional Safety. International Organization for Standardization (2011)
46. ISO/IEC: 27034-2:2015 - Information technology – Security techniques – Application security – Part 2: Organization normative framework. http://goo.gl/D8EClR. Online; accessed May 2016
47. ISO/IEC: 9646-7:1995 - Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements. http://goo.gl/9WtcAy. Online; accessed May 2016
48. ISO/IEC: ISO/IEC 15408-1:2009 - Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model. http://goo.gl/F0A5aT. Online; accessed May 2016
49. ITEA: DIAMONDS - Development and Industrial Application of Multi-Domain Security Testing Technologies. https://itea3.org/project/diamonds.htm. Online; accessed September 2016
50. Izerrouken, N., Kai, O.S.Y., Pantel, M., Thirioux, X.: Use of formal methods for building qualified code generator for safer automotive systems. In: Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness & Safety, pp. 53–56 (2010)
51. Kaur, M., Singh, P., et al.: Performance evaluation of v2vcommunication by implementing security algorithm in vanet. In: Advances in Computing and Information Technology, pp. 757–763. Springer (2012)
52. Kornecki, A.J., Zalewski, J.: Safety and security in industrial control. In: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, p. 77. ACM (2010)
53. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., et al.: Experimental security analysis of a modern automobile. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 447–462. IEEE (2010)
54. Kounga, G., Walter, T., Lachmund, S.: Proving reliability of anonymous information in vanets. IEEE Transactions on Vehicular Technology **58**(6), 2977–2989 (2009)
55. Leinmüller, T., Buttyan, L., Hubaux, J.P., Kargl, F., Kroh, R., Papadimitratos, P., Raya, M., Schoch, E.: Sevecom-secure vehicle communication. In: IST Mobile and Wireless Communication Summit, LCA-POSTER-2008-005 (2006)
56. Leinmüller, T., Schmidt, R.K., Held, A.: Cooperative position verification-defending against roadside attackers 2.0. In: Proceedings of 17th ITS World Congress, pp. 1–8 (2010)

57. Lindlar, F., Zimmermann, A.: A code generation tool for embedded automotive systems based on finite state machines. In: Proceedings of 6th IEEE International Conference on Industrial Informatics, pp. 1539–1544 (2008)
58. Macher, G., Stolz, M., Armengaud, E., Kreiner, C.: Filling the gap between automotive systems, safety, and software engineering. e & i Elektrotechnik und Informationstechnik **132**(3), 142–148 (2015)
59. Malip, A., Ng, S.L., Li, Q.: A certificateless anonymous authenticated announcement scheme in vehicular ad hoc networks. Security and Communication Networks **7**(3), 588–601 (2014)
60. MathWorks: Embedded Coder. http://it.mathworks.com/products/embedded-coder/index.html. Online; accessed May 2016
61. MathWorks: Simulink - Simulation and Model-Based Design. http://it.mathworks.com/help/simulink/. Online; accessed May 2016
62. Miller, C., Valasek, C.: A survey of remote automotive attack surfaces. Black Hat USA (2014)
63. Navet, N., Simonot-Lion, F.: In-vehicle communication networks-a historical perspective and review. Industrial Communication Technology Handbook, 2nd ed **96**, 1204–1223 (2013)
64. Nolte, T., Hansson, H., Bello, L.L.: Automotive communications-past, current and future. In: Proceedings of 10th IEEE Conference on Emerging Technologies and Factory Automation, vol. 1, p. 8. IEEE (2005)
65. OMG: Systems Modeling Language. http://www.omgsysml.org/. Online; accessed May 2016
66. OMG: The UML Profile for MARTE: Modeling and Analysis of Real-Time and Embedded Systems. http://www.omgmarte.org/. Online; accessed May 2016
67. Open Garages: Car Hackers 2014.Owner Manual. http://goo.gl/H1Byqn. Online; accessed May 2016
68. Papadimitratos, P., Buttyan, L., Holczer, T.S., Schoch, E., Freudiger, J., Raya, M., Ma, Z., Kargl, F., Kung, A., Hubaux, J.P.: Secure vehicular communication systems: design and architecture. Communications Magazine, IEEE **46**(11), 100–109 (2008)
69. Popov, P.: Preliminary interdependency analysis (PIA): Method and tool support. In: E. Troubitsyna (ed.) Software Engineering for Resilient Systems, *Lecture Notes in Computer Science*, vol. 6968, pp. 1–8. Springer Berlin Heidelberg (2011)
70. Prasad, K.V., Broy, M., Krueger, I.: Scanning advances in aerospace & automobile software technology. Proceedings of the IEEE **4**(98), 510–514 (2010)
71. Pretschner, A., Broy, M., Kruger, I.H., Stauner, T.: Software engineering for automotive systems: A roadmap. In: 2007 Future of Software Engineering, pp. 55–71 (2007)
72. Razzaque, M., Salehi, A., Cheraghi, S.M.: Security and privacy in vehicular ad-hoc networks: survey and the road ahead. In: Wireless Networks and Security, pp. 107–132. Springer (2013)
73. Robinson-Mallett, C.: Coordinating security and safety engineering processes in automotive electronics development. In: Proceedings of the 9th Annual Cyber and Information Security Research Conference, pp. 45–48 (2014)
74. Sagstetter, F., Lukasiewycz, M., Steinhorst, S., Wolf, M., Bouard, A., Harris, W.R., Jha, S., Peyrin, T., Poschmann, A., Chakraborty, S.: Security challenges in automotive hardware/software architecture design. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 458–463. EDA Consortium (2013)
75. Schroeder, J., Berger, C., Herpel, T.: Challenges from integration testing using interconnected hardware-in-the-loop test rigs at an automotive oem: An industrial experience report. In: Proceedings of the First International Workshop on Automotive Software Architecture, pp. 39–42. ACM (2015)
76. Sikora, E., Tenbergen, B., Pohl, K.: Industry needs and research directions in requirements engineering for embedded systems. Requirements Engineering **17**(1), 57–78 (2012)
77. Sommerville, I.: Software Engineering. International computer science series. ed: Addison Wesley. (2004)
78. Sporer, H., Macher, G., Armengaud, E., Kreiner, C.: Incorporation of model-based system and software development environments. In: Proceedings of 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 177–180 (2015)

79. Studnia, I., Nicomette, V., Alata, E., Deswarte, Y., Kaâniche, M., Laarouchi, Y.: Survey on security threats and protection mechanisms in embedded automotive networks. In: Proceedings of 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W), pp. 1–12 (2013)
80. Toeppe, S., Bostic, D., Ranville, S., Rzemien, K.: Automatic code generation requirements for production automotive powertrain applications. In: Proceedings of the IEEE International Symposium on Computer Aided Control System Design, pp. 200–206 (1999)
81. Toom, A., Izerrouken, N., Naks, T., Pantel, M., Ssi-Yan-Kai, O.: Towards reliable code generation with an open tool: Evolutions of the gene-auto toolset. In: Proceedings of 5th International Congress and exhibition ERTS2 (2010)
82. Voget, S.: Autosar and the automotive tool chain. In: Proceedings of the Conference on Design, Automation and Test in Europe, pp. 259–262. European Design and Automation Association (2010)
83. Wiedersheim, B., Sall, M., Reinhard, G.: Sevecomsecurity and privacy in car2car ad hoc networks. In: Proceedings of 9th International Conference on Intelligent Transport Systems Telecommunications (ITST), pp. 658–661 (2009)
84. Wolff, C., Brink, C., Httger, R., Igel, B., Kamsties, E., Krawczyk, L.: Automotive Software Development with AMALTHEA. In: Practice and Perspectives, p. 432 (2015)