embeddedworld2018
Exhibition&Conference
... it's a smarter world

Safety & Security Testing of Cooperative Automotive Systems

Dominique Seydel, Gereon Weiss
Application Architecture Design & Validation
Fraunhofer ESK
Munich, Germany
{dominique.seydel, gereon.weiss}@esk.fraunhofer.de

Daniela Pöhn, Sascha Wessel
Secure Operating Systems
Fraunhofer AISEC
Garching, Germany
{daniela.poehn, sascha.wessel}@aisec.fraunhofer.de

Franz Wenninger
Design, Test & System Integration
Fraunhofer EMFT
Munich, Germany
franz.wenninger@emft.fraunhofer.de

***Abstract*— Cooperative behavior of automated traffic participants is one next step towards the goals of reducing the number of traffic fatalities and optimizing traffic flow. The notification of a traffic participant's intentions and coordination of driving strategies increase the reaction time for safety functions and allow a foresighted maneuver planning. When developing cooperative applications, a higher design complexity has to be handled, as components are distributed over heterogeneous systems that interact with a varying timing behavior and less data confidence. In this paper, we present a solution for the development, simulation and validation of cooperative automotive systems together with an exemplary development flow for safety and security testing.**

Keywords*— *automotive safety; cooperative applications; security testing; validation; autonomous systems; ITS

I. INTRODUCTION

In comparison to the development of traditional ADAS functions, testing and simulation of connected applications have to consider the interaction of heterogeneous systems that are distributed within a wireless networked architecture. As the communication link is more unreliable in contrast to common input sensors, the application has to cope with varying timing behavior and less data confidence. However, the higher complexity in the development process of cooperative applications is justified by several advantages. They arise from the aspect that foreign traffic participants are no longer solely observed from the outside to predict their behavior but they give insights on their status, intentions and their involvement in cooperative maneuvers. This results in an increased reliability of predicted vehicle movements, which in turn can be used for safety functions and allow an increased reaction time of safety mechanisms. In terms of driving comfort, the traffic participant's cooperation allows a foresighted maneuver planning.

By the current state of available tools, the development, test and certification of autonomous systems is complex, costly in terms of time and equipment, potentially hazardous and often incomplete. For instance, if it comes to complex applications that require a distributed consensus, e.g. Merging Assistance [1], an application distributed among various foreign entities has to be validated. Thus, it appears that development and simulation environments are not yet ready to rapidly develop prototypes of cooperative driving functions.

Therefore, we provide an approach for an integrated testing environment that can cover the whole innovation cycle for prototype development of cooperative automotive systems. Incorporating safety and security aspects, it starts from the design of applications over simulation to integrating and validating the respective prototypes. Thus, our approach supports the whole development process for prototyping and testing cooperative functions.

The following Chapter II gives an overview on an efficient approach for the development of cooperative applications. Further aspects of the simulation and testing phase are discussed in Chapter III. The current scope of software analysis is presented in Chapter IV and a secured deployment process in the following Chapter V. We conclude our work with Chapter VI and provide a brief outlook to next steps.

II. RAPID APPLICATION DEVELOPMENT

A. Application Development Cycle

The testbed combines several aspects of Vehicle-to-X (V2X) application development life cycle. It covers development steps beginning from application design,

continuous integration into simulation environments, testing environments over tools performing functional and security analyses, up to a secure deployment and update process. The application development flow of the innovation- and testbed concept is shown in Fig. 1. It is also designed to only use single aspects in a building-block style when developing innovative applications.

The testing layer consists of several testing methods that are specific for each step in the development process and include several simulation environments, integration testing and field tests. The testbed provides the ezCar2x® framework, described in [2], that allows testing connected applications within a simulation environment, using network and traffic simulation as well as integration of hardware-in-the-loop, e.g. Road Side Units (RSUs). Another main feature of the testbed is a combined simulation and field testing approach, where virtual and real traffic participants can be tested in a synchronized environment.

Additional analysis tools enable to examine the developed application. On the one hand, our DANA tool ("Description and Analysis of Networked Applications") for functional validation can be used iteratively in every testing step [3]. For the integration testing of the application, the analysis toolbox provides application security testing methods in order to detect software vulnerabilities in an early development stage. Using static application security testing (SAST) as well as dynamic application security testing (DAST) allows quick analyses during integration testing in order to detect potential software vulnerabilities.

Finally, the application is built and subsequently signed within the software repository and pushed to the update server, which is part of the back end. The update server again signs the application and deploys it to V2X devices.

B. *Application Design*

For the initial development step of designing a cooperative driving function, the testing environment comprises interfaces to common automotive modelling tools, like Matlab Simulink or ADTF. The deployed application uses the ezCar2x® framework, an ETSI ITS (Intelligent Transport Systems) compliant communication stack, which can either run on real communication hardware or on a virtual node within a network simulation. Furthermore, application security testing can be conducted with static and dynamic methods.

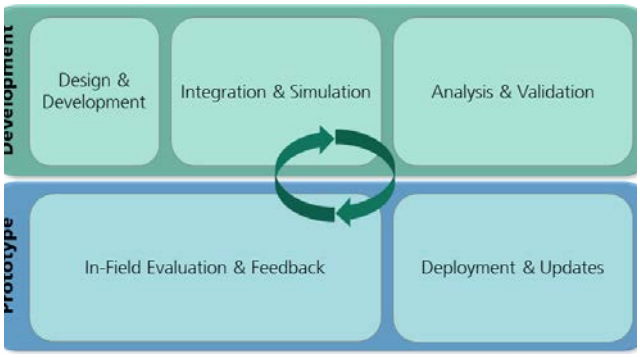

Fig. 1 Application Development Cycle using the Testbed Services

If enhanced safety mechanisms are required for the intended application, state-of-the-art software methods, e.g. graceful degradation strategies [4] or model-based communication [5], can be incorporated into the application model within this development step as well. For example, a connected application with safety-critical functionality, as Platooning, strongly depends on Quality of Service (QoS) parameters of the communication link. Our safety function for Resilient Control uses these QoS parameters, such as the current Packet Loss Rate (PLR), to decide which degradation mode is sufficient, e.g. readjusting the distance to the vehicle ahead. The safety mechanisms for resilient control are developed as generic component and can be integrated into common automotive software architectures, as AUTOSAR, AUTOSAR Adaptive and further concepts. Also existing architectures from non-safety domains like infotainment, as developed by the GENIVI Alliance, can be integrated to handle the unreliability of the communication link.

Another aspect that is getting more relevant for application design and validation are so-called Plastic Architectures. Parts of an application can be distributed over several entitites. For example, in the case of a Collision Warning application [6] this includes the interaction of the originating, the warning and optionally edge or cloud components form the overall function. As the specific architecture may frequently change, the architectures change depending on the context, thus becoming formable or plastic. Also, in future parts of the application can be dynamically relocated during runtime, e.g. from cloud over edge to in-vehicle components. Thereby, the system boundaries dynamically change depending on the current communication relations. Although, there are concepts to solve the underlying network aspects [7], these runtime conditions have already to be covered within the design phase of the specific applications.

III. Simulation & Testing

One goal of simulation and testing for cooperative automated driving is to achieve a fail-operational behavior of the application, even when the context information is unconfident. Therefore, the coverage of the testcases used within the simulation environment and during virtual testing should be as realistic and as comprehensive as possible. This is reached by (automatically) defining reference scenarios and generate variations out of them, e.g. by stochastic variations [8].

One of the parameter variations is the realistic behavior of the communication channel during a certain driving scenario. Therefore, a network simulation tool, e.g. ns-3 or OMNET++, and a traffic simulation tool, e.g. SUMO, VTD or CarMaker, are integrated into the simulation environment. Our testbed could also be integrated with other microscopic and macroscopic traffic simulation tools, as each of them has advantages when testing a specific connected application.

A. *Simulation Environment*

The suggested concept combines three different simulation aspects into one integrated simulation environment.

The first component is a traffic simulator that is used to model and run driving test cases on a realistic road network.

The second component is a network simulation tool for evaluating applications under real communication conditions. For the heterogeneous use of common vehicular communication technologies, e.g. 802.11p, 4G or LTE, the ezCar2x® framework provides additional network layer components. The network simulation tool also facilitates interfaces to control traffic simulation and integration hardware-in-the-loop tests or vehicle-in-the-loop tests (as for Virtual Platooning), e. g. including RSUs.

The third component of the testbed is for test control. Traces from all simulation components are monitored and analyzed within the test control component. For ensuring the security of cooperative systems, testing covers white-, gray-, and black-box approaches (e.g. Data-Flow Analysis, Fuzzing or Penetration Testing). In order to validate the applications, test cases have to reach full coverage and should therefore be generated (semi-)automatically for each application.

B. Integrated and Hybrid Simulation

As already described in Chapter II.A, the application implementation is deployed on each virtual V2X node within the network simulation environment. Together with the ezCar2x® Framework each virtual node can be equipped with developed applications and also with V2X communication ability. Hence, its interaction with other nodes can be simulated as realistically as possible.

The interaction between all virtual nodes is realized using a virtual wireless channel. Thereby, we consider the specific characteristics of each communication technology by using individual channel models, e.g. dedicated models for ITSG5, LTE or 5G. This virtual wireless channel can also be used to integrate real hardware into the simulation by using a channel proxy and creating a mirror node for each hardware component within the network simulation.

The network simulation is coupled with a macroscopic traffic simulation for large scale traffic scenarios, e.g., to test security mechanisms for V2X messages, and with a microscopic traffic simulation for smaller driving scenarios, e.g. Cooperative Merging [6] or Platooning. The coupling via a control interface is needed to synchronize the behavior of communication nodes and traffic participants in each simulation for the given driving scenario.

The environment can be extended with hybrid simulation capabilities by including hardware-in-the-loop. A RSU comprising an application, e.g. Smart Lighting, can be integrated into the simulation loop, by connecting it to the wireless channel interface of the simulation environment. The RSU can again interact with further communication hardware, e.g. test vehicles that are in communication range. The RSU can also be connected with sensors, that are integrated into the testing environment and which can provide status data to generate event messages, e.g. Decentralized Environmental Notification Messages (DENMs).

C. Sensor Integration

The effectiveness of a connected applications' simulation depends on how realistic the input data for a certain driving scenario is. The input data from distributed sources, e.g. the status data within Cooperative Awareness Messages (CAMs) from other vehicles or roadside sensors, have to be synchronized during recording and replay phase. Synchronization is required to setup the intended driving conditions for the application that are to be tested. To achieve realistic input data, it is beneficial to have sensors integrated in an early development step, through Hardware-in-the-Loop (HiL) or recorded input data stream integration.

The realistic sensor data can also be used to develop algorithms for sensor analysis, timing and improvement of machine learning processes. In addition, to develop tamper-proof algorithms it is advantageous to have a realistic behavior of sensor data available in order to avoid manipulation or attacks with sensor hardware or sensor data.

Within our simulation environment, vehicle sensors and infrastructure sensors are integrated as components in ezCar2x® via generic sensor interfaces. When recording test data, the real sensors can easily be included into the synchronized recording process. The same setup can be used during field tests, where infrastructure sensors usually are integrated into RSUs and thereby provide their environment data. Thus, virtual, hybrid and integration testing can be carried out with low effort.

IV. SOFTWARE ANALYSIS

In each step of the development process it is beneficial to perform additional analyses to get detailed knowledge of the overall system and the applications behavior for debugging, monitoring, security, and validation purposes. In this chapter, we give an overview on our methods and tools for software analysis.

An exemplary flow of the development process for an application is shown in Fig. 2. The application under development can be prototyped as implemented source code or as a software model to be further developed and optimized within the testbed.

A. Monitoring and Functional Validation

For software validation and verification, model-based techniques are advantageous during the design and integration phase. Our DANA platform [3], an open and modular environment based on Eclipse, is a tool built for specifying and analyzing networked applications. For this purpose, the specified valid behavior of the application is described as a layered reference model. This model provides a basis for further model-based development steps. On the one hand, it can be used for various transformations of behavior models, e.g., for generating test cases or code for running simulations. On the other hand, it can be used for static analyses to check conformance to modeling guidelines, metrics for interfaces, and the compatibility of behavior models. The model-based approach also allows a quick integration of new messages sources, e.g. additional communication protocols or wireless channels. Furthermore, the DANA tool can be used for verifying and validating software interface behavior, as messages in these interfaces can contain complex data and include intricate interactions.

In our proposed testbed we use DANA as a central monitoring tool to have all the status, debug and behavior

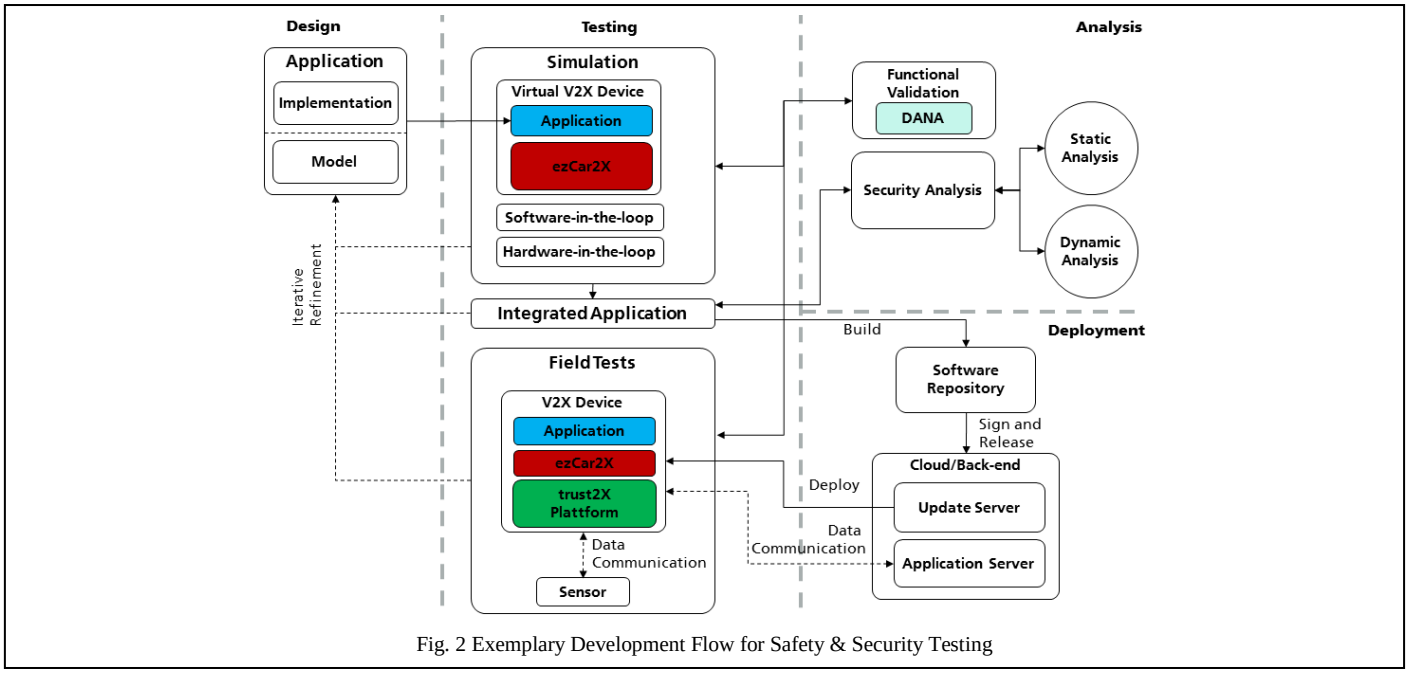

Fig. 2 Exemplary Development Flow for Safety & Security Testing

information, the error messages and timing data centrally available from each component. This aggregation helps to simplify and to speed up the debugging process during development and runtime. Further validation checks can be applied on this collected data, as described in the previous paragraph.

B. Application Security Testing

By integrating static as well as dynamic testing methods into the development process, applications can be tested against a broad spectrum of software vulnerabilities, as outlined in [9]. Detecting vulnerabilities in early development stages is crucial, as it prevents the necessity of expensive software patching post-release. Therefore, we propose a work flow, where application security testing is part of continuous integration.

Static code analysis as part of SAST allows inspecting program behavior without actually running the program. A large amount of tools for programming languages that are typically used within the V2X domain are available for detecting software requirements being violated, e.g. erroneous program behavior, reachability or dead code. At that, many tools are specifically designed to detect program flaws that lead to potential vulnerabilities, e.g. buffer overflows.

SAST depends on the application source code and vulnerability specification as input, where the latter defines what kind of vulnerabilities should be detected by the tool. The static analysis then runs fully automated and outputs a report of the detected potential vulnerabilities.

Available tools can be applied directly on source code as well as on binaries and bytecode. The applied methods differ in efficiency and effectiveness depending on the underlying approach. Broadly, applied methods can be divided into lexical scanning and data or control flow analysis.

While SAST has shortcomings, e.g. detection of vulnerabilities in authentication, access control or cryptographic protocols, as well as uncovering flaws in the security design, DAST can tackle these problems. Furthermore, it provides deeper analysis and can imitate attack scenarios. Since applications are executed within this testing method, DAST depends on predefined input data. These inputs either represent specific test cases or are randomly chosen.

Dynamic code analysis is a white-box testing approach. This means that the internal structure of an application, e.g. source code or intermediate representation, is available and can be leveraged for analyzing the application.

Monitoring of function calls by hooking into security critical functions, e.g. system calls, is one typical testing method. In addition, function parameter analysis is applied, inspecting the input-output relations of function calls. More sophisticated methods like dynamic taint analysis are able to analyze execution paths an attacker may use to exploit an application. This method can also be applied if source code of an application is not available.

All of these methods aim at detecting potential vulnerabilities that occur during runtime of the application. This allows verifying alarms that have been reported by SAST. Furthermore, it can be used as preparation for penetration testing, as it provides potential entry points for actual attacks.

V. Secure Deployment and Prototyping

To rapidly transfer the developed application into a prototype, as needed for field tests, an integrated deployment process is beneficial. The testbed offers an integrated solution containing a secure V2X platform, a continuous integration workflow, secure deployment and update processes and a communication link to back end or cloud platforms.

Novel cooperative functions can be integrated with ezCar2x® into secure ITS prototype devices. These build upon trust2X [10], a hardened platform that includes hardware- and software-based security in order to isolate and protect

processes and data of cooperative driving functions from other operating systems (e. g. AUTOSAR), functional modules and communication interfaces (e. g. backend communication for secure software updates and app deployment).

The hardened V2X platform trust2X is a hardened Linux-based operating system that provides hardware- and software-based security features for target devices of V2X applications. Its main goal is the secure isolation of different software entities that run concurrently on top of a single Linux kernel. ezCar2x® can run in a Linux container, isolated from other guest operating systems (e.g. AUTOSAR) and other functional modules or services (e.g. TLS, VPN). By isolating software entities, each entity is protected against compromised or malfunctioning software that runs in a different container. Therefore, safety- and security-critical functions can run within an isolated environment, unaffected by failure or compromise of separate entities in the system.

Each commit to the central software repository automatically triggers a build of the application under development. While each build is self-testing, SAST and DAST are triggered automatically. Static security testing tools are applied either directly on source code, or on compilation products such as bytecode or binaries. Therefore, either single commits can be tested using light-weight analysis tools, as well as individual software modules or the complete application. Static application testing tools can run without user interaction and provide a report, which lists each finding of potential vulnerabilities. DAST tools might require user interaction depending on the applied testing method.

In case a potential vulnerability has been found the continuous integration server alerts the development team. Based on the review of the testing reports either further security tests can be applied, or the developer commits a patch in order to eradicate the vulnerability. If the application has been successfully tested and no potential vulnerabilities were found, the continuous integration server triggers a merge request, or directly merges with master. Subsequently, the application is build and signed.

VI. Conclusion

We provided an approach for an integrated testing environment that covers the whole development process for prototyping and testing cooperative functions. Incorporating safety and security aspects starting from the design phase, the complex task of simulation and cooperative applications and several testing steps have been described. Finally, a software solution for the validation and deployment of the prototypes was presented, which makes tools available for the whole development cycle. Thereby, a toolkit is provided that is intended to rapidly bring an idea for a connected application into a prototype with a decreased investment risk.

In future, all testbed services described in the previous chapters could also be made available as an online web-service. For this purpose, a next step of the described solution is to enable access to configurable or pre-configured simulations via an online service. On this website, users could remotely control simulation parameters, define new scenarios and get qualified evaluation results. This ongoing development addresses the rapid development of connected applications by abstracting technology know–how and increasing time-to-market speed. Thus, innovators and developers can concentrate on the actual function and idea of the intended application and are able to experience, improve, and validate their solution in early stages prior to competitors.

Acknowledgment

This project was partially funded by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology within the High Performance Center Secure Networked Systems.

References

[1] Ntousakis, I. A., Nikolos, I. K., & Papageorgiou, M. (2017). Cooperative Vehicle Merging on Highways-Model Predictive Control (No. 17-00930).

[2] Roscher, K., Bittl, S., Gonzalez, A. A., Myrtus, M., and Jiru, J. (2014). ezCar2X: Rapid-Prototyping of Communication Technologies and Cooperative ITS Applications on Real Targets and Inside Simulation Environments, In: 11th Conference Wireless Communication and Information. vwh, pp. 51 – 62.

[3] Drabek, C., Weiss, G. (2017) DANA - Description and Analysis of Networked Applications. In: International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CuBES), pp. 71-80.

[4] Schleiss P., Drabek C., Weiss G., Bauer B. (2017) Generic Management of Availability in Fail-Operational Automotive Systems. In: Tonetta S., Schoitsch E., Bitsch F. (eds) Computer Safety, Reliability, and Security. SAFECOMP 2017.

[5] Moradi-Pari, E., Mahjoub, H. N., Kazemi, H., Fallah, Y. P., and Tahmasbi-Sarvestani, A. (2017). Utilizing Model-Based Communication and Control for Cooperative Automated Vehicle Applications. IEEE Transactions on Intelligent Vehicles.

[6] Zhang, R., Cao, L., Bao, S., & Tan, J. (2017). A method for connected vehicle trajectory prediction and collision warning algorithm based on V2V communication. International Journal of Crashworthiness, 22(1), 15-25.

[7] An, X., et al. (2017) On end to end network slicing for 5G communication systems. In: Transactions on Emerging Telecommunications Technologies, 28. Jg., Nr. 4.

[8] Damm W., Heidl P. (Hrsg) (2017) Positionspapier und Roadmap zu „Hochautomatisierte Systeme: Testen, Safety und Entwicklungsprozesse“, SafeTRANS e. V. http://www.safetrans-de.org/de/Aktuelles/?we_objectID=2, Access: 18.1.2018.

[9] Eckert, Claudia. (2017) Cybersicherheit beyond 2020!. In: Informatik-Spektrum, 40. Jg., Nr. 2, pp. 141-146.

[10] Waidner M. (2018) Safety und Security. In: Neugebauer R. (eds) Digitalisierung. Springer Vieweg, Berlin, Heidelberg