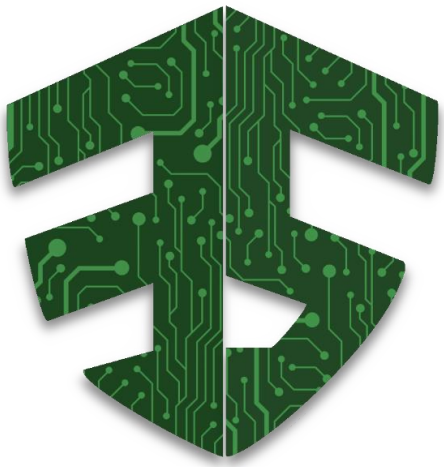


Verification and Validation of Automated Systems' Safety and Security

Identified gaps and limitations of the V&V methods listed in D3.1

Document Type	Report
Document Number	D3.3
Primary Author(s)	Enrico Ferrari (Rulex), Rupert Schlick (AIT)
Document Date	2021-04-26
Document Version	1.2 Final
Dissemination Level	Public (PU)
Reference DoA	2021-02-26
Project Coordinator	Behrooz Sangchoolie, behrooz.sangchoolie@ri.se, RISE Research Institutes of Sweden
Project Homepage	www.valu3s.eu
JU Grant Agreement	876852

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.

Disclaimer

Project Overview

Manufacturers of automated systems and the manufacturers of the components used in these systems have been allocating an enormous amount of time and effort in the past years developing and conducting research on automated systems. The effort spent has resulted in the availability of prototypes demonstrating new capabilities as well as the introduction of such systems to the market within different domains. Manufacturers of these systems need to make sure that the systems function in the intended way and according to specifications which is not a trivial task as system complexity rises dramatically the more integrated and interconnected these systems become with the addition of automated functionality and features to them.

With rising complexity, unknown emerging properties of the system may come to the surface making it necessary to conduct thorough verification and validation (V&V) of these systems. Through the V&V of automated systems, the manufacturers of these systems are able to ensure safe, secure and reliable systems for society to use since failures in highly automated systems can be catastrophic.

The high complexity of automated systems incurs an overhead on the V&V process making it time-consuming and costly. VALU3S aims to design, implement and evaluate state-of-the-art V&V methods and tools in order to reduce the time and cost needed to verify and validate automated systems with respect to safety, cybersecurity and privacy (SCP) requirements. This will ensure that European manufacturers of automated systems remain competitive and that they remain world leaders. To this end, a multi-domain framework is designed and evaluated with the aim to create a clear structure around the components and elements needed to conduct the V&V process through identification and classification of evaluation methods, tools, environments and concepts that are needed to verify and validate automated systems with respect to SCP requirements.

In VALU3S, 12 use cases with specific safety, security and privacy requirements will be studied in detail. Several state-of-the-art V&V methods will be investigated and further enhanced in addition to implementing new methods aiming for reducing the time and cost needed to conduct V&V of automated systems. The V&V methods investigated are then used to design improved process workflows for V&V of automated systems. Several tools will be implemented supporting the improved processes which are evaluated by qualification and quantification of safety, security and privacy as well as other evaluation criteria using demonstrators. VALU3S will also influence the development of safety, security and privacy standards through an active participation in related standardisation groups. VALU3S will provide guidelines to the testing community including engineers and researchers on how the V&V of automated systems could be improved considering the cost, time and effort of conducting the tests.

VALU3S brings together a consortium with partners from 10 different countries, with a mix of *industrial partners* (24 partners) from automotive, agriculture, railway, healthcare, aerospace and industrial automation and robotics domains as well as leading *research institutes* (6 partners) and *universities* (10 partners) to reach the project goal.

Consortium

RISE RESEARCH INSTITUTES OF SWEDEN AB	RISE	Sweden
STAM SRL	STAM	Italy
FONDAZIONE BRUNO KESSLER	FBK	Italy
KNOWLEDGE CENTRIC SOLUTIONS SL - THE REUSE COMPANY	TRC	Spain
UNIVERSITA DEGLI STUDI DELL'AQUILA	UNIVAQ	Italy
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO	ISEP	Portugal
UNIVERSITA DEGLI STUDI DI GENOVA	UNIGE	Italy
CAMEA, spol. s r.o.	CAMEA	Czech
IKERLAN S. COOP	IKER	Spain
R G B MEDICAL DEVICES SA	RGB	Spain
UNIVERSIDADE DE COIMBRA	COIMBRA	Portugal
VYSOKE UCENI TECHNICKE V BRNE - BRNO UNIVERSITY OF TECHNOLOGY	BUT	Czech
ROBOAUTO S.R.O.	ROBO	Czech
ESKISEHIR OSMANGAZI UNIVERSITESI	ESOGU	Turkey
KUNGLIGA TEKNISKA HOEGSKOLAN	KTH	Sweden
STATENS VAG- OCH TRANSPORTFORSKNINGSINSTITUT	VTI	Sweden
UNIVERSIDAD DE CASTILLA - LA MANCHA	UCLM	Spain
FRAUNHOFER GESELLSCHAFT ZUR FOERDERUNG DER ANGEWANDTEN FORSCHUNG E.V.	FRAUNHOFER	Germany
SIEMENS AKTIENGESELLSCHAFT OESTERREICH	SIEMENS	Austria
RULEX INNOVATION LABS SRL	RULEX	Italy
NXP SEMICONDUCTORS GERMANY GMBH	NXP-DE	Germany
PUMACY TECHNOLOGIES AG	PUMACY	Germany
UNITED TECHNOLOGIES RESEARCH CENTRE IRELAND, LIMITED	UTRCI	Ireland
NATIONAL UNIVERSITY OF IRELAND MAYNOOTH	NUIM	Ireland
INOVASYON MUHENDISLIK TEKNOLOJI GELISTIRME DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI	IMTGD	Turkey
ERGUNLER INSAAT PETROL URUNLERI OTOMOTIV TEKSTIL MADENCILIK SU URUNLER SANAYI VE TICARET LIMITED STI.	ERARGE	Turkey
OTOKAR OTOMOTIV VE SAVUNMA SANAYI AS - OTOKAR AS	OTOKAR	Turkey
TECHY BILISIM TEKNOLOJILERI DANISMANLIK SANAYI VE TICARET LIMITED SIRKETI - TECHY INFORMATION TECHNOLOGIESAND CONSULTANCY LIMITED COMPANY	TECHY	Turkey
ELECTROTECNICA ALAVESA SLe	ALDAKIN	Spain
INTECS SOLUTIONS SPA	INTECS	Italy
LIEBERLIEBER SOFTWARE GMBH	LLSG	Austria
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY GMBH	AIT	Austria
E.S.T.E. SRL	ESTE	Italy
NXP SEMICONDUCTORS FRANCE SAS	NXP-FR	France
BOMBARDIER TRANSPORTATION SWEDEN AB	BT	Sweden
QRTECH AKTIEBOLAG	QRTECH	Sweden
CAF SIGNALLING S.L	CAF	Spain
MONDRAGON GOI ESKOLA POLITEKNIKOA JOSE MARIA ARIZMENDIARRIETA S COOP	MGEP	Spain
INFOTIV AB	INFOTIV	Sweden
BERGE CONSULTING AB	BERGE	Sweden

Executive Summary

Within the scope of WP3 - *Design of SCP (Safety, Cybersecurity, and Privacy) V&V (Verification and Validation) methods for automated systems*, D3.3 analyses the gaps and limitations of the state-of-the-art methods presented in D3.1. The analysis is done with a double approach: on one side, the 53 methods are analysed, and their general limitations are reviewed; on the other side, the use cases are reviewed in order to find the gaps that prevent or limit the application of V&V in those scenarios.

For a better evaluation, gaps and limitations have been divided into different categories:

- **Accuracy**: limitations in the accuracy of the method.
- **Scalability and computational**: limitations in applying the method to larger problems.
- **Deployment**: issues regarding the deployment of the method in real-world contexts.
- **Learning curve**: limitations related to high-level skills required to apply the method.
- **Lack of automation**: issues with allowing the method to be executed without human intervention.
- **Reference environment**: limitations regarding the reference environment where the method can be applied.
- **Cost:** limitations related to the high cost needed to use the method.

The identification of gaps and limitations will guide the work of the next WP3 efforts, in particular regarding the development of improved, new or married (i.e. obtained by combining two already existing techniques) methods to bridge the gaps. This work is going to be performed in Task 3.3.

Contributors

Enrico Ferrari	RULEX	Jose Luis de la Vara	UCLM
Arturo García	UCLM	Luis Alonso	TRC
Marco Bozzano	FBK	Stefano Tonetta	FBK
Davide Ottonello	STAM	Massimo Nazaria	FBK
Hamid Ebadi	INFOTIV	Martin Karlsberg	INFOTIV
Thanh Bui	RISE	Joakim Rosell	RISE
Tomas Vojnar	BUT	Ales Smrcka	BUT
Giorgio Malaguti	ESTE	Maurizio Lo Piccolo	ESTE
Michele Mingozzi	ESTE	Nicola Caselli	ESTE
Marie Farrell	NUIM	Fredrik Warg	RISE
Matt Luckcuck	NUIM	Peter Folkesson	RISE
Rosemary Monahan	NUIM	Mateen Malik	RISE
Jorge Valero	UCLM	Fabio Patrone	UNIGE
Markus Borg	RISE	Giovanni Gaggero	UNIGE
Giovanni Giachetti	UCLM	Alessandro Fausto	UNIGE
Bernhard Fischer	SIEMENS	Martin Matschnig	SIEMENS
Metin Ozkan	ESOGU	Christoph Sohrmann	FRAUNHOFER
Silvia Mazzini	INTECS	Juan Santana	FRAUNHOFER
Leire Exteberria	MGEP	Gabriel Pachiana	FRAUNHOFER
Joseba Andoni Agirre	MGEP	Alper Kanak	ERARGE
Aitor Agirre	MGEP	Salih Ergün	ERARGE
Georgios Giantamidis	UTRCI	Sercan Tanrıseven	ERARGE
Stylianos Basagiannis	UTRCI	Rupert Schlick	AIT
Christoph Schmittner	AIT	Sankar Sathyamoorthy	QRTECH
Gürol Çokünlü	OTOKAR	Muhammet Saral	OTOKAR
Ömer Şahabaş	OTOKAR	Johnny Öberg	KTH
David Fürcho	NXP	Michael Philipp	NXP
Iñigo Elguea	ALDAKIN	Thomas Bauer	FRAUNHOFER
Santiago Gonzalez	ALDAKIN	Bernd Bredehorst	PUMACY
Ugur Yayan	IMTGD	Pierre Kirisci	PUMACY
Martin Skoglund	RISE	Mustafa Karaca	IMTGD

Reviewers

Giovani Giachetti	UCLM	2021-03-24
Jose Luis de la Vara	UCLM	2021-03-25, 2021-04-19
Mateen Malik	RISE	2021-03-29
Erwin Kristen	AIT	2021-03-29
Tomas Vojnar	BUT	2021-04-01
Behrooz Sangchoolie	RISE	2021-04-20, 2021-04-26

Revision History

Version	Date	Author (Affiliation)	Comment
0.1	2020-12-29	Enrico Ferrari (RULEX)	Initial Draft Template
0.2	2021-02-17	Enrico Ferrari (RULEX) et al.	Most contributions from methods gaps
0.3	2021-02-25	Enrico Ferrari (RULEX) et al.	New contributions on methods gaps
0.4	2021-03-02	Enrico Ferrari (RULEX)	Written Chapter 1 (Introduction)
0.5	2021-03-12	Enrico Ferrari (RULEX) et al.	Integrated new contributions and general description.
0.6	2021-03-18	Enrico Ferrari (RULEX) et al.	General part and integrated new contributions from partners.
0.7	2021-03-22	Enrico Ferrari (RULEX) et al.	Deliverable ready for internal review
0.8	2021-04-15	Enrico Ferrari (RULEX) et al.	Deliverable ready for intermediary approval
1.0	2021-04-23	Enrico Ferrari (RULEX) et al.	Deliverable ready for final approval
1.1	2021-04-26	Behrooz Sangchoolie (RISE)	Review of the final draft while making minor formatting changes.
1.2	2021-04-26	Behrooz Sangchoolie (RISE)	Final version to be submitted.

Table of Contents

List of Figures

List of Tables

Acronyms

AADL	Architecture Analysis and Design Language
ADAS	Advanced Driver Assistance Systems
ADS	Automated Driving System
ASAM	Association for Standardization of Automation and Measuring Systems
CAD	Computer Aided Design
CD	Continuous development
CI	Continuous integration
CPS	Cyber-Physical System
CPU	Central Processing Unit
DAE	Digital Asset Exchange file format
ESMINI	Environment Simulator Minimalist
FDD	Failure Detection and Diagnosis
FLA	Failure Logic Analysis
FMEA	Failure Modes and Effects Analysis
FPGA	Field-Programmable Gate Array
FRET	Formal Requirements Elicitation Tool
FT	Fault Tree
FTA	Fault Tree Analysis
HISA	Human Interaction Safety Analysis
HMI	Human Machine Interaction
HRI	Human Robot Interaction
IC	Integrated Circuit
IDE	Integrated Development Environment
IDS	Intrusion Detection System
KLOC	Thousands of Lines Of Code
LTL	Linear Temporal Logic
MBD	Model-Based Design
MBSA	Model Based Safety Analysis
MIAI	Model-Implemented Attack Injection
MIFI	Model-Implemented Fault Injection
ML	Machine Learning
MoMuT	Model-based Mutation Testing

MQTT	Message Queuing Telemetry Transport
NMT	Neuromuscular Transmission
PLC	Programmable Logic Controller
PTC	Post Tetanic Count
QEMU	Quick EMUlator
RAM	Random Access Memory
RNG	Random Number Generator
ROS	Robot Operating System
RTL	Register Transfer Level
RTR	Run-Time Reconfiguration
SCP	Safety, Cybersecurity and Privacy
SDR	Software-Defined Radio
SEM	Soft-Error Mitigation
SMT	Satisfiability Modulo Theories
SoC	System on Chip
SQL	Structured Query Language
STL	Standard Triangle Language
SUMO	Simulation of Urban Mobility
SUT	System Under Test
SW	Software
TOF	Train Of Four
UVM	Universal Verification Methodology
V&V	Verification and Validation
VCC	A Verifier for Concurrent C
WP	Work Package
WPM	Weak Process Model
WSN	Wireless Sensor Network

Chapter 1 Introduction

As the use and complexity of automated systems are growing, system manufacturers and component suppliers require methods that can help them to confirm that the SCP requirements of the systems are satisfied. This is necessary so that the systems can be deemed dependable and secure. Even if much effort has been spent in research and development in the field of Verification & Validation (V&V), the rising complexity of automated systems makes very difficult to prove that systems will behave as they were planned to do. As illustrated in Figure 1.1, the frequency of cyber-physical disasters has increased in particular starting from 2005 affecting many sectors like automotive and health. The main point is that threats have evolved from mainly software-based ones to a complex combination of software and the hardware levels. So, the approach to respond to attacks on SCADA systems, sensor networks, automation software and control units should inevitably be holistic: no component can be considered separately. This increased the level of the challenge to ensure the SCP requirements are met in automated systems. The result is that many applications in different sectors are still not covered by the currently available methods. Many factors cause this technological gap, ranging from computational issues to the lack of a sufficient degree of accuracy. One of the goals of the VALU3S project is to develop new V&V methods that can overcome current limitations.

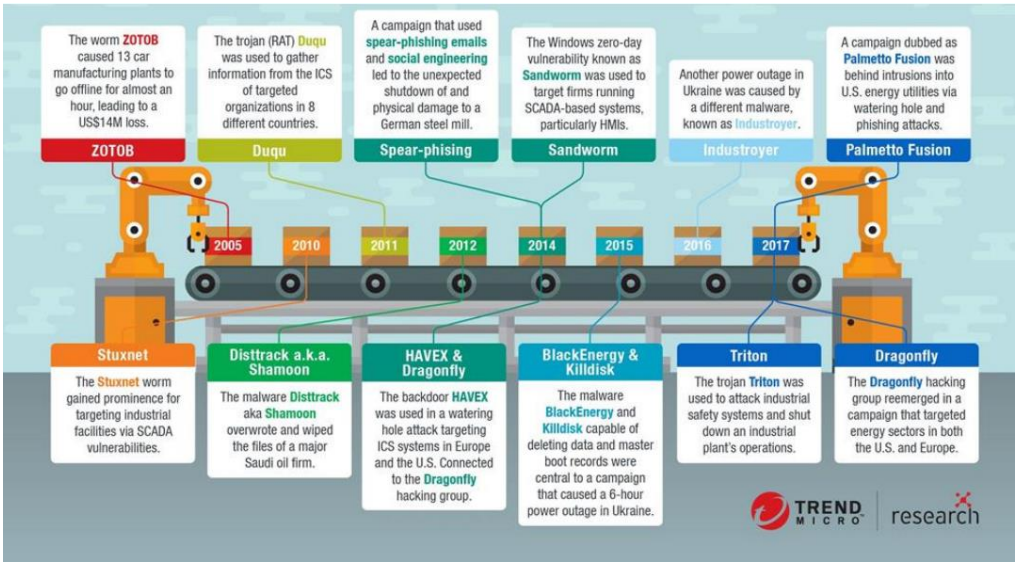


Figure 1.1: Evolvement of security threats and risks over the past decades

In particular, the aim of WP3 (Design of SCP V&V methods for automated systems) is to create a set of reference methods for V&V of automated systems. The WP3 objective is reached with three subsequent steps:

1. Studying the currently available state of the art V&V methods.
2. Identifying gaps and limitations in methods found in 1.
3. Improve, combine existing or develop completely new methods according to the limitations highlighted in 2.

All the activities of WP3 are carried out with a strong connection with the VALU3S use cases to ensure that all the developments respond to actual needs deriving from real world scenarios. Task 3.2 deals with step 2 of this process, i.e. the identification of gaps and limitations in the currently available methods. This deliverable reports the results of this analysis. It is worth noting that on one hand the analysis is general since it takes into account the methods independently from their specific application in the project. For this reason, the gap analysis has included also gaps and limitations that are out of the scope of the project because they were considered valuable information. On the other hand, the focus and the way the topics are presented are influenced by the specific applications considered in the project. So, the list of gaps should not be regarded as an exhaustive one and within the list more focus is given to gaps and limitations of interest for the project's goals.

Since the methods are extensively explained in Deliverable 3.1 (D3.1) [1], the focus here is not on the methods' functionality but more on what is lacking for their wider and better employment. Therefore, the reader should refer to D3.1 for further details on the methods.

This deliverable relates to other VALU3S' deliverables that either provide input or will use its results as a basis for their development:

- D1.1 (Description of use cases as well as scenarios) [2] and D1.2 (SCP requirements as well as identified test cases) [3] provide the use case scenarios to consider for identifying gaps of the V&V methods in D3.2.
- D3.1 (V&V methods for SCP evaluation of automated systems) [1] reviews the state of the art that is used to identify the gaps of currently available methods.
- D3.4 (Initial description of methods designed to improve the V&V process), D3.5 (Interim description of methods designed to improve the V&V process), and D3.6 (Final description of methods designed to improve the V&V process) will largely base their work on the insights provided in this deliverable.
- D5.1 (Initial demonstration plan and a list of evaluation criteria) [4] is providing details on use cases and their evaluation criteria, as well as the currently used approaches to V&V.

The following chapters introduce the background of the deliverable (Chapter 2) and present the identified gaps in V&V methods (Chapter 3) as well as some overview and statistics about them (Chapter 4). Then, the identified gaps in tools (Chapter 5) and the insights deriving from the use cases (Chapter 6) will be reported. Chapter 7 closes the deliverable with some conclusions.

Chapter 2 Background

The identification of gaps and limitations in V&V has proceeded in two different but complementary directions (see Figure 2.1). On one side, gaps and limitations have been searched in the methods from a more theoretical point of view. This approach will be referred to as *bottom up*. Regarding this approach, the focus is both on the methods themself and on the tools supporting the methods, pointing out the issues related to the implementation. On the other side, the use cases are analysed to highlight applications that cannot be carried out with current methods and therefore require improvements or completely new methods. This top-down approach will allow us to ensure that developments are done with a proper level of generality and theoretical soundness and, at the same time, that they are oriented to solve real world problems.

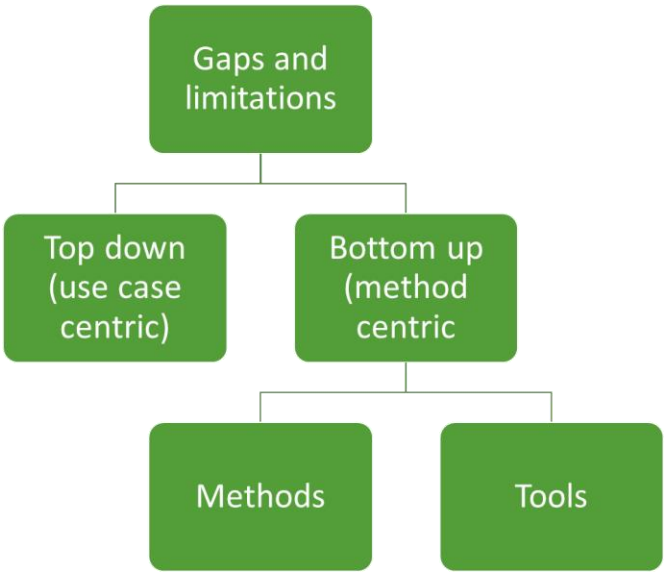


Figure 2.1: Approach followed for the analysis of gaps in V&V methods

To better identify and group gaps, different categories have been defined:

- **Accuracy**: the method has some limitations regarding its accuracy. For example, the method is not reliable enough for some critical application where it is employed.
- **Scalability and Computational**: the method requires too many computational resources (time and/or memory) and therefore can be applied only to limited/simplified scenarios.
- **Deployment**: the method presents some problems when deployed in real-world contexts. For example, there is lack of proper tools or there are issues in integrating the methods with other platforms.
- **Learning curve**: to be properly used, the method requires high-level technical skills that are not easy to find.
- **Lack of automation**: the method is not fully automatic, i.e., it requires heavy intervention, such as tuning, by human users. As a consequence, the V&V process could become long and error prone.

- **Reference environment**: the method works only for some reference environment, e.g., simulation. There are no warranties that findings are still valid for other environments.
- **Cost:** using the method requires huge investments in terms of e.g., hardware, software, and human resources.

All the methods presented in D3.1 have been analysed in Chapter 3 of this deliverable according to the criteria defined above. For ease of reading, the same groups of methods defined in D3.1 have been considered:

- Injection-based V&V
  - Fault injection
  - Attack injection
- Simulation
- Testing
- Runtime verification
- Formal verification
  - Formal source code verification
  - General formal verification
- Semi-formal analysis
  - SCP-focused semi-formal analysis
  - General semi-formal analysis
- System-type-focused V&V

Moreover, all the gaps have been identified by a unique label in order to allow related deliverables (in particular D3.4, D3.5, and D3.6) to reference the gaps that have been addressed during the project's activities. Regarding this, it is worth noting that the list presented here includes more gaps than those that will be addressed during the project. All the identified gaps have been mentioned in this deliverable for reporting purposes, but it is likely that not all of them are going to be addressed either because they have more structural limitations or because they are out of the scope of the applications considered in VALU3S. Moreover, it is worth noting that the work of identifying gaps and limitations in methods will continue after Task 3.2 within other tasks both at a technological level (in particular in Task 3.3) and from a use case perspective (in particular, in WP5).

Chapter 3 Gaps and Limitations in V&V Methods

This chapter reports gaps and limitations of V&V methods presented in D3.1 [1]. The presentation of gaps and limitations follows the same structure as the state-of-the-art review in D3.1: methods are divided in categories and sub-categories and for each method the different types of gaps and limitations are pointed out. It may happen that no gaps or limitations are present for a method and a category (for example *Vulnerability and Attack Injection* does not have any reported gaps or limitations as regards scalability and computational issues). In this case, usually a short explanation is reported to clarify why no significant gap or limitation is mentioned. All the gaps and limitations are labelled by a proper id and decomposed in "atomic gaps" as much as possible, so that it is easier to understand which gaps have been addressed in VALU3S Task 3.3. For this reason, it often happens that several gaps or limitations are reported in each category for a single method. Each gap or limitation is labelled with the prefix "GAPM-" (gap of the method) followed by a three digits-code identifying the method (e.g. "MIA" is the code for Model-Implemented Attack Injection) and by a sequential number.

3.1 Injection-Based V&V

This group of methods focuses on introducing certain characteristics in a system, providing a certain type of input, or triggering certain events, to confirm that the system behaves suitably under the corresponding conditions. Two types of injection are considered: attack injection and fault injection.

3.1.1 Attack Injection

3.1.1.1 Model-Implemented Attack Injection

Name of the method: **Model-Implemented Attack Injection**
Short description In this method, the attacks (which are special types of faults) are injected in the model of the System Under Test (SUT) [MIA01]. MATLAB and LabVIEW are examples of tools used to build such system models. This method is used to verify and validate the system's capability to handle attacks. The attack handling includes mechanisms to detect and prevent intrusions [MIA02]. This type of attack injection method is used for the system's evaluation at early design stages.
Limitations
Functionality [GAPM-MIA01] The method can be improved by adding techniques such as pre-injection analysis and post-injection analysis [MIA03] [MIA04] [MIA05] [MIA06] to reduce the number of the tests and still get the same or improved results in terms of time, cost and effort. [GAPM-MIA02] Although there are many generic attack models, attack models often have to be adapted for the target system.
Accuracy

[GAPM-MIA03] The accuracy of the method depends on the accuracy of the modelled attacks and systems.
Scalability and computational [GAPM-MIA04] Exhaustive attack injection or full system monitoring may require intense computational resources depending on the complexity of the target system and its environment.
Deployment [GAPM-MIA05] The model-implemented attack injection method is not feasible for final implementations of systems. [GAPM-MIA06] The method must be adapted for the simulation tool environment used, e.g., MATLAB toolboxes and MATLAB versions used.
Learning curve [GAPM-MIA07] The method requires knowledge and skills regarding the simulation tool environment, e.g., MATLAB/SIMULINK skills.
Lack of automation [GAPM-MIA08] The configuration and result analysis are done manually.
Reference environment [GAPM-MIA09] This method is only applicable for the simulation environment.
Costs The cost of implementing this method is minimal because there is no hardware needed to execute the tests by using this method. [GAPM-MIA10] Software such as MATLAB/SIMULINK is not opensource and needs investments. [GAPM-MIA11] Test cost increases when new attack models are implemented. [GAPM-MIA12] There is also some cost involved in terms of time when conducting the test. For example, exhaustive attack injection or full system monitoring increases the cost of verification and validation.
Standards The requirements of the standards which this method fulfils are ISO-TC22-SC32-WG11_N0613_ISO_SAE_DIS_21434_(E), NIST 800, IEC 62443, SAE J3061, IEC TR 63069, IEC TR 63074, ISO TR 22100-4, ISO 24089
References
• [MIA01] B. Sangchoolie, P. Folkesson, and J. Vinter, “A study of the interplay between safety and security using model-implemented fault injection,” in 2018 14th Eur. Dep. Comp. Conf. (EDCC). IEEE, 2018, pp. 41–48.

- [MIA02] B. Sangchoolie, P. Folkesson, Pierre Kleberger and J. Vinter, "Analysis of Cybersecurity Mechanisms with respect to Dependability and Security Attributes," in 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops.
- [MIA03] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock and J. Haid, "Efficient fault emulation using automatic pre-injection memory access analysis," 2012 IEEE International SOC Conference, Niagara Falls, NY, 2012, pp. 277-282.
- [MIA04] B. Sangchoolie, F. Ayatolahi, R. Johansson and J. Karlsson, "A Comparison of Inject-on-Read and Inject-on-Write in ISA-Level Fault Injection," 2015 11th European Dependable Computing Conference (EDCC), Paris, 2015, pp. 178-189.
- [MIA05] Czeck, Edward W. and Daniel P. Siewiorek. "Observations on the Effects of Fault Manifestation as a Function of Workload." IEEE Trans. Computers 41 (1992): 559-566.
- [MIA06] Folkesson P., Karlsson J. (1999) Considering Workload Input Variations in Error Coverage Estimation. In: Hlavička J., Maehle E., Pataricza A. (eds) Dependable Computing — EDCC-3. EDCC 1999. Lecture Notes in Computer Science, vol 1667. Springer, Berlin, Heidelberg.

3.1.1.2 Simulation-Based Attack Injection at System-level

Name of the method: **Simulation-Based Attack Injection at System-level**

Short description

Simulation-based Attack Injection at System-level provides an opportunity of injecting attacks on the system level. Different parts of the system and their interconnections can be verified and validated by using this technique. The complete system behaviour can be analysed when a certain sub-system is under the influence of attacks. While conducting field tests could be costly and sometimes life-threatening, simulation-based tests provide a wide range of advantages, such as lower testing costs, adaptation of tests to a variety of traffic scenarios, and avoiding the life-threatening situations.

This method could span over various tools such as SUMO (Simulation of Urban Mobility) [SAI02], CARLA (autonomous driving simulator) [SAI03] and VEINS (VEhicles In Network Simulation) [SAI04] allowing different aspects of the system to be evaluated.

Limitations

Functionality

[GAPM-SAI01] More features can be added in the method functionality such as improving the representativeness of both attack models and simulated systems.

Accuracy

[GAPM-SAI02] Modelling of a system in a simulation environment might not accurately represent the real system in a real environment. So, the final V&V activities are recommended to be performed on a real system.

Scalability and computational

[GAPM-SAI03] Exhaustive attack injection and full system monitoring may require intense computational resources.

[GAPM-SAI04] Selecting complex scenarios for attack injection may require high computational resources.
Deployment [GAPM-SAI05] The method must be adapted for the simulation tools, such as SUMO, CARLA, and VEINS, and the tools versions used.
Learning curve [GAPM-SAI06] The method requires knowledge of simulators and skills to use.
Lack of automation [GAPM-SAI07] The test configuration and result analysis are done manually.
Reference environment [GAPM-SAI08] This method is only applicable for the simulation environment.
Costs [GAPM-SAI09] The cost could depend on the type of simulator required for the V&V of the specific system requirements, e.g., proper system test for CARLA could cost a bit in terms of hardware and processing. [GAPM-SAI10] Attack injection is time consuming which could increase the cost depending on how we want to do it, e.g., running exhaustive attack injection experiments. [GAPM-SAI11] The time could be reduced if experiments could be run in parallel on the expense of the hardware increase, which increases the cost factor.
Standards No relevant gap or limitation has been identified. The requirements of the standards which this method fulfils are ISO 26262, IEC 62061, IEC TR 63074, ISO PAS 21448, ISO 13849, IEC 61508, ISO/IEC TR 24028:2020, ISO/IEC WD 23053
References
• [SAI01] Eduardo dos Santos et al., "Towards a Simulation-based Framework for the Security Testing of Autonomous Vehicles" • [SAI02] Michael Behrisch, Laura Bieker et al., "SUMO – Simulation of Urban Mobility, An Overview", Institute of Transportation Systems, German Aerospace Center, Rutherfordstr. 2, 12489 Berlin, Germany. • [SAI03] Alexey Dosovitskiy, German Ros et al., "CARLA: An Open Urban Driving Simulator". • [SAI04] Veins - Vehicles in Network Simulation, http://veins.car2x.org

3.1.1.3 Vulnerability and Attack Injection

Name of the method: **Vulnerability and Attack Injection**
Short description

The method consists of injecting realistic vulnerabilities in the target component and mounting attacks through the exploitation of the injected vulnerabilities. The goal is to evaluate how the system where the target component is inserted, including existing security components (i.e. intrusion detection systems, security personnel), can cope with the attacks (i.e., the target component is not under evaluation; the rest of the system is) [VAI1], [VAI2].
Limitations
Functionality [GAPM-VAI01] The injection of the different types of vulnerabilities is complex, which makes it difficult to implement tools with a rich variety of vulnerability types. [GAPM-VAI02] The number and variety of possible attacks could be limited. [GAPM-VAI03] Heavily dependent on the programming language of the target application. [GAPM-VAI04] Needs to access the source code of the application or system.
Accuracy [GAPM-VAI05] Limitations in accuracy are inherent to the coverage limitations of vulnerability injection.
Scalability and computational [GAPM-VAI06] The number of vulnerabilities and the time needed to perform an injection campaign depends on the target component. In any case, even when the target component is large and complex the method generally scales well [VAI3].
Deployment [GAPM-VAI07] Lack of mature tools outside academia.
Learning curve [GAPM-VAI08] The learning curve is relatively steep if the practitioner is not knowledgeable in cybersecurity.
Lack of automation [GAPM-VAI09] Concerning the injection of vulnerabilities, the degree of automation is similar to software fault injection approaches. The attack step is fully automatic.
Reference environment [GAPM-VAI10] Requires a prototype or a real system. [GAPM-VAI11] Target components must be exposed to possible attacks, typically target components should be accessible through the Internet.
Costs

No relevant gap or limitations has been identified. The number of vulnerabilities and the time needed to perform an injection campaign depends on the target component. In any case, this is generally not a problem and does not represent a significant cost in time needed to perform a campaign.
Standards No relevant gap or limitations has been identified. The method can be used in the context of the following standards: IEC TR 63074, ISO/IEC TR 24028:2020, ISO/IEC 27001
References
• [VAI1] J. Fonseca, N. Seixas, M. Vieira, and H. Madeira, "Analysis of Field Data on Web Security Vulnerabilities", IEEE Transactions on Dependable and Secure Computing, accepted for publication in 2014. • [VAI2] J. Fonseca, M. Vieira, and H. Madeira, "Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection", IEEE Transactions on Dependable and Secure Computing, accepted for publication in 2014. • [VAI3] Elia, I., Fonseca, J., Vieira, M., "Comparing SQL Injection Detection Tools Using Attack Injection: An Experimental Study", The 21st annual International Symposium on Software Reliability Engineering (ISSRE 2010), November, 2010.

3.1.2 Fault Injection

3.1.2.1 Fault Injection in FPGAs

Name of the method: **Fault injection in FPGAs**
Short description The objective is the evaluation of possible results of fault injection and their propagation in an FPGA-based Hardware Platform.
Limitations
Functionality [GAPM-FIF01] This technique is based on Healing Core approach which is subject to errors as well. Some of encountered errors may not be healed by simply resetting and/or rebooting the entire FPGA. Thus, it is not clear that this method has an important advantage considering the time and cost dedicated to it.
Accuracy [GAPM-FIF02] It is up to test design to achieve a high-level accuracy in these tests. By modifying data residing in the configuration bits of an FPGA, the result of this fault injection can be observed. However, there are many possible combinations of flipping configuration bits. Thus, overall accuracy of this test is directly related to coverage of all possible scenarios.
Scalability and computational

No relevant gap or limitation has been identified since this technique can easily be scaled to any FPGA so that there is not any expected scalability issue. Required computational resources to carry out this test are considerably limited. Hence, it can be performed easily using a PC or Workstation, but it is less certain that it can be carried out by processor cores embedded in the FPGA.

Deployment

[GAP-FIF03] The implementation of this test can be carried out mostly during the development phase. Soft-Error Mitigation (SEM)-cores to detect errors and uses Run-Time Reconfiguration (RTR) techniques to correct Single- and Multiple-Event Upsets (bit-flips) in the FPGA's configuration memory. Further, it has a classification system that can report and initiate appropriate countermeasures for some faults. Thus, it can be used to implement self-repairing functionality in an FPGA system.

Learning curve

[GAPM-FIF04] The test approach used requires some solid background in FPGAs. Thus, it requires high-level skills for implementation.

Lack of automation

No relevant gap or limitation has been identified since the tests can be executed either manually or in an automatic fashion once required test software is developed.

Reference environment

No relevant gap or limitation has been identified since the tests can be implemented in prototype stage and at the operation environment (TRL-7). However, the software for using the method in the operating environment is not currently developed.

Costs

No relevant gap or limitation has been identified since the tests currently require a workstation, related software and human labor dedicated to this test.

Standards

[GAP-FIF05] Current Safety standards is not in favour of using FPGAs in the design because of the risk for bit-flips and changed functionality.

References

- [FIF1] E. Kyriakakis, K. Ngo, J. Öberg, "Mitigating Single-Event Upsets in COTS SDRAM using an EDAC SDRAM Controller", In Proc. of 2017 IEEE Nordic Circuits and Systems Conference (NorCAS-2017), Linköping, Sweden, Oct 24-25, 2017.
- [FIF2] E. Kyriakakis, K. Ngo, J. Öberg, "Implementation of a Fault-Tolerant, Globally-Asynchronous-Locally-Synchronous, Inter-Chip NoC Communication Bridge on FPGAs", In Proc. of 2017 IEEE Nordic Circuits and Systems Conference (NorCAS-2017), Linköping, Sweden, Oct 24-25, 2017.

- [FIF3] K. Ngo, T. Mohammadat, J. Öberg, "Towards a Single Event Upset Detector Based on COTS FPGA", In Proc. of 2017 IEEE Nordic Circuits and Systems Conference (NorCAS-2017), Linköping, Sweden, Oct 24-25, 2017.
- [FIF4] Öberg, J., Robino, F., "A NoC System Generator for the Sea-of-Cores Era", In Proc. of FPGAWorld 2011, Copenhagen, Stockholm, Munich, September, 2011, ACM Digital Libraries.

3.1.2.2 Interface Fault Injection

Name of the method: **Interface fault injection**
Short description Injection of invalid inputs at the interface of software components (OS calls, APIs, services or any type of interface defined in the component) in order to evaluate the behaviour of the target component in the presence of such faulty inputs [IFI01], [IFI02], [IFI04]. Current tool (bBOXRT - https://git.dei.uc.pt/cnl/bBOXRT) is available to inject fault in web services [IFI03], [IFI04].
Limitations
Functionality [GAPM-IFI01] The classification of results is highly dependent on the detailed knowledge of the system under testing and the target component (i.e., the component in which the faults are injected at interface level). [GAPM-IFI01] The quality of generated workloads often limits the disclosure of robustness problems.
Accuracy [GAPM-IFI03] The method is accurate in the sense that it injects invalid parameters that allow accurate assessment of the target component robustness (i.e., behaviour of the target component in the presence of invalid inputs). In any case, the domain of invalid parameters could be quite large, which means that only a sample of invalid parameters is tested (coverage problem).
Scalability and computational Does not present limitations since the faults are determined by the interface of the target component(s) and the method is not affected by the scale of the system under test.
Deployment No relevant gap or limitation has been identified since the method is fully dynamic and can only be used when a prototype or a deployed version of the system under test is available.
Learning curve [GAPM-IFI05] The method is easy to use since the tool (bBOXRT) is fully automatic, but the method requires the knowledge of the software architecture of the system under test and the interface of the target component to allow correct result analysis and interpretation.

Lack of automation No relevant gap or limitation has been identified since the method is fully automatic.
Reference environment [GAPM-IFI06] Requires a prototype or a real system.
Costs [GAPM-IFI07] The number of faults and the time needed to perform an injection campaign depends on the number of parameters and the domains of the parameters of the interface of the target component. In any case, this is generally not a problem and does not represent a significant cost.
Standards No relevant gap or limitation has been identified since the method is fully automatic. As a specific type of fault injection, the method can be used in the context of the standards ISO 26262, IEC 62061, IEC TR 63074, ISO PAS 21448, ISO 13849, IEC 61508, ISO/IEC TR 24028:2020, ISO/IEC WD 23053.
References
• IFI1] N. Laranjeiro, M. Vieira and H. Madeira, "Experimental Robustness Evaluation of JMS Middleware," 2008 IEEE International Conference on Services Computing, Honolulu, HI, 2008, pp. 119-126, doi: 10.1109/SCC.2008.129. • [IFI2] J. Cámara, R. de Lemos, N. Laranjeiro, R. Ventura and M. Vieira, "Robustness-Driven Resilience Evaluation of Self-Adaptive Software Systems," in IEEE Transactions on Dependable and Secure Computing, vol. 14, no. 1, pp. 50-64, 1 Jan.-Feb. 2017, doi: 10.1109/TDSC.2015.2429128. • [IFI3] N. Laranjeiro, M. Vieira, & H. Madeira, A robustness testing approach for SOAP Web services. J Internet Serv Appl3, 215–232 (2012).https://doi.org/10.1007/s13174-012-0062-2 • [IFI4] N. Laranjeiro, M. Vieira and H. Madeira, "A Technique for Deploying Robust Web Services," in IEEE Transactions on Services Computing, vol. 7, no. 1, pp. 68-81, Jan.-March 2014, doi: 10.1109/TSC.2012.39.

3.1.2.3 *Model-Based Fault Injection for Safety Analysis*

Name of the method: **Model-Based Fault Injection for Safety Analysis**
Short description In model-based fault injection, user-specified faults may be (automatically) injected into a system model to generate an extended model that specifies the behavior of the system in presence of faults. Fault injection can be performed manually or using a library of predefined failure modes. The extended model can be used to perform safety analysis activities, such as FTA and FMEA.
Limitations
Functionality

[GAPM-MBI01] Tool support usability can be improved, e.g., editing and customization of fault libraries.
Accuracy [GAPM-MBI02] The method is accurate, as long as the system model and the fault specifications are accurate. It is recommended to perform V&V activities to verify that the system model and the extended system model accurately represent the intended behaviour.
Scalability and computational [GAPM-MBI03] The automated analysis, in presence of a high number of faults, may be subject to the state-explosion problem, impacting the effectiveness of verification.
Deployment [GAPM-MBI04] The method is well supported by tools such as xSAP and COMPASS However, these tools cover the phase of architectural design and the highest levels of behavior specifications; they are not suitable for deployment to the target HW.
Learning curve [GAPM-MBI05] The method requires model-based design skills, as well as safety expertise.
Lack of automation No relevant gap or limitation has been identified, since the model extension is fully automated. The method requires the user to specify a set of faults, taken from the fault library, to be injected into the system model. The library of faults is fully general; if needed, additional user-defined faults can be added.
Reference environment No gap is envisaged since the method may be applied at development level in TRL6-7 environments and prototypes.
Costs The xSAP [MBI1] and OCRA [MBI2] tools are freely available for non-commercial applications. The COMPASS tool [MBI3, MBI4] is freely available for ESA member states.
Standards [GAPM-MBI06] The method is conceived to fulfil safety standards (e.g., ECSS, EN 50129, SAE-ARP-4754, SAE-ARP-476). It has to be verified in VALU3S if the method is suitable for the specific medical and agriculture domains (e.g., CEI EN 62304, ISO 14971, IEC 61508, ISO 26262).
References

- [MBI1] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli and G. Zampedri. The xSAP Safety Analysis Platform. In Proceedings of TACAS 2016. Eindhoven, The Netherlands, April 2-8, 2016.
- [MBI2] A. Cimatti, M. Dorigatti, S. Tonetta. OCRA: A tool for checking the refinement of temporal contracts. Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE), pages 702-705, IEEE, 2013.
- [MBI3] M.Bozzano, A.Cimatti, J.-P.Katoen, V. Y.Nguyen, T.Noll and M.Roveri. Safety, Dependability, and Performance Analysis of Extended AADL Models. The Computer Journal, 54(5):754-775, 2011.
- [MBI4] M. Bozzano, A. Cimatti, J.-P. Katoen, P. Katsaros, K. Mokos, V.Y. Nguyen, T. Noll, B. Postma and M. Roveri. Spacecraft Early Design Validation using Formal Methods. Reliability Engineering & System Safety 132:20-35. December 2014.

3.1.2.4 Model-Implemented Fault Injection

Name of the method: **Model-implemented Fault Injection**
Short description In this method, the faults are injected in the model of the system under test (SUT) [MIF01]. MATLAB and LabVIEW are examples of tools used to build such system models. This method is used to verify and validate the system's capability to handle faults. The fault handling includes attributes such as fault detection, correction, or fallback with or without the fault handling mechanisms implemented. This type of fault injection method is used for the system's evaluation at early design stages [MIF02].
Limitations
Functionality [GAPM-MIF01] The method can be improved by adding techniques such as pre-injection analysis and post-injection analysis [MIF03] [MIF04] [MIF05] [MIF06] to reduce the number of the tests and still get the same or improved results in terms of time, cost and effort. Pre-injection analysis is done before any fault injection experiments are performed while post-injection uses the results of previous fault injection experiments. [GAPM-MIF02] Adding more fault models will increase the functionality of the method.
Accuracy [GAPM-MIF03] The accuracy of the method depends on the accuracy of the modelled faults and systems. Since the model of the system might not accurately represent the real system in a real environment, V&V activities (acceptance tests) are recommended to be performed in a later development stage.
Scalability and computational [GAPM-MIF04] Exhaustive fault injection or full system monitoring may require a lot of computational resources depending on the complexity of the target system and its environment.

Deployment [GAPM-MIF05] The model-implemented fault injection method is not feasible for final implementations of systems. [GAPM-MIF06] The method must be adapted for the simulation tool environment used, e.g., MATLAB toolboxes and MATLAB versions used.
Learning curve [GAPM-MIF07] The method requires knowledge and skills regarding the simulation tool environment, e.g., MATLAB/SIMULINK skills.
Lack of automation [GAPM-MIF08] The configuration of fault injection campaigns and result analysis are done manually.
Reference environment [GAPM-MIF09] This method is only applicable for the simulation environment.
Costs [GAPM-MIF10] Software such as MATLAB/SIMULINK is not opensource and needs investments. [GAPM-MIF11] There is also some cost involved in terms of time when conducting model implemented fault injection. For example, exhaustive fault injection or full system monitoring increases the cost of verification and validation.
Standards No relevant gap or limitation has been identified. The requirements of the standards which this method fulfils are ISO 2626, IEC 62061, IEC TR 63074, ISO PAS 21448, ISO 13849, IEC 61508, ISO/IEC TR 24028:2020, ISO/IEC WD 23053.
References
• [MIF01] R. Svenningsson, J. Vinter, H. Eriksson, and M. Törngren, “Modifi: A model-implemented fault injection tool,” in Proc. of the 29th Int. Conf. on Computer Safety, Reliability, and Security, ser. SAFECOMP’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 210–222. • [MIF02] P. Folkesson, F. Ayatolahi, B. Sangchoolie, J. Vinter, M. Islam, and J. Karlsson, “Back-to-back fault injection testing in model-based development,” in Computer Safety, Reliability, and Security, 2015. • [MIF03] J. Grinschgl, A. Krieg, C. Steger, R. Weiss, H. Bock and J. Haid, "Efficient fault emulation using automatic pre-injection memory access analysis," 2012 IEEE International SOC Conference, Niagara Falls, NY, 2012, pp. 277-282. • [MIF04] B. Sangchoolie, F. Ayatolahi, R. Johansson and J. Karlsson, "A Comparison of Inject-on-Read and Inject-on-Write in ISA-Level Fault Injection," 2015 11th European Dependable Computing Conference (EDCC), Paris, 2015, pp. 178-189.

- [MIF05] Czeck, Edward W. and Daniel P. Siewiorek. "Observations on the Effects of Fault Manifestation as a Function of Workload." IEEE Trans. Computers 41 (1992): 559-566.
- [MIF06] Folkesson P., Karlsson J. (1999) Considering Workload Input Variations in Error Coverage Estimation. In: Hlavička J., Maehle E., Pataricza A. (eds) Dependable Computing — EDCC-3. EDCC 1999. Lecture Notes in Computer Science, vol 1667. Springer, Berlin, Heidelberg

3.1.2.5 *Simulation-Based Fault Injection at System-level*

Name of the method: **Simulation-Based Fault Injection at System-level**
Short description Simulation-based Fault Injection at System-level provides an opportunity of injecting faults on the system level [SFI01]. The complete system behaviour can be analysed when a certain sub-system is under the influence of faults. This method could span over various tools such as SUMO (Simulation of Urban Mobility) [SFI02], CARLA (autonomous driving simulator) [SFI03], and VEINS (vehicular network simulator) [SFI04], allowing different aspects of the system to be evaluated.
Limitations
Functionality [GAPM-SFI01] More features can be added in the method functionality such as improving the representativeness of both fault models and simulated systems. [GAPM-SFI02] Moreover, pre-injection and post-injection techniques [SFI05] [SFI06] [SFI07] can also be used to improve the functionality.
Accuracy [GAPM-SFI03] Modelling of a system in a simulation environment might not accurately represent the real system in a real environment. So, the V&V activities are also recommended to be performed on a real system.
Scalability and computational [GAPM-SFI04] Exhaustive fault injection, full system monitoring may require a lot of computational resources. [GAPM-SFI05] Selecting complex and more realistic scenarios for fault injection may also require high computational resources. [GAPM-SFI06] The simulation tools with intensive 3D simulations (which is in line with this method implementation) often requires 3D rendering and that cannot be accomplished without GPUs and increased processing power. This poses challenges on scalability and computational power.
Deployment [GAPM-SFI07] The method must be adapted for the simulation tools, such as SUMO, CARLA, and VEINS, and the tools versions used.

Learning curve [GAPM-SFI08] The method requires knowledge of traffic simulators and skills to use.
Lack of automation [GAPM-SFI09] The test configuration and result analysis are done manually.
Reference environment [GAPM-SFI10] This method is only applicable for the simulation environment.
Costs [GAPM-SFI11] That could depend on the type of simulator required for the V&V of the specific system requirements, e.g., proper system test for CARLA could costs a bit in terms of hardware and processing. [GAPM-SFI12] Fault injection is also time consuming which could increase the cost depending on how we want to do it, e.g., running exhaustive fault injection experiments. [GAPM-SFI13] The time could be reduced if experiments could be run in parallel on the expense of the hardware increase which increases the cost factor.
Standards No relevant gap or limitation has been identified. The requirements of the standards which this method fulfils are ISO 26262, IEC 62061, IEC TR 63074, ISO PAS 21448, ISO 13849, IEC 61508, ISO/IEC TR 24028:2020, ISO/IEC WD 23053
References
• [SFI01] M.-C. Hsueh, T.K. Tsai, and R.K. Iyer, "Fault Injection Techniques and Tools," Computer, vol. 40, no. 4, pp. 75-82, Apr. 1997. • [SFI02] S. Jha et al., "AVFI: Fault Injection for Autonomous Vehicles," in Proc. 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 55–56. • [SFI03] Michael Behrisch, Laura Bieker et al., "SUMO – Simulation of Urban Mobility, An Overview", Institute of Transportation Systems, German Aerospace Center, Rutherfordstr. 2, 12489 Berlin, Germany. • [SFI04] Veins - Vehicles in Network Simulation, http://veins.car2x.org • [SFI05] B. Sangchoolie, F. Ayatolahi, R. Johansson and J. Karlsson, "A Comparison of Inject-on-Read and Inject-on-Write in ISA-Level Fault Injection," 2015 11th European Dependable Computing Conference (EDCC), Paris, 2015, pp. 178-189. • [SFI06] Czeck, Edward W. and Daniel P. Siewiorek. "Observations on the Effects of Fault Manifestation as a Function of Workload." IEEE Trans. Computers 41 (1992): 559-566. • [SFI07] Folkesson P., Karlsson J. (1999) Considering Workload Input Variations in Error Coverage Estimation. In: Hlavička J., Maehle E., Pataricza A. (eds) Dependable Computing — EDCC-3. EDCC 1999. Lecture Notes in Computer Science, vol 1667. Springer, Berlin, Heidelberg.

3.1.2.6 Software-Implemented Fault Injection

Name of the method: **Software-Implemented Fault Injection**
Short description The method [FIN1] emulates representative faults through the insertion of errors in computer systems and/or components using software means. The errors inserted must reproduce similar conditions observed in the field when real faults of different types occur. Existing tools (e.g., ucXception [FIN2]) basically emulate two types of faults: hardware transient faults (bit flips) and software faults (most frequent types of bugs found in field studies [FIN3]).
Limitations
Functionality [GAPM-FIN01] As it often happens in injection approaches, fault coverage is a limitation. It may happen that the injected faults are not a representative sample of all possible faults. [GAPM-FIN02] The used fault models should be realistic and represent faults that the system may experience. It could be difficult to prove that is the case. [GAPM-FIN03] Inherent intrusiveness of the tool as the fault injection tool may skew the results.
Accuracy [GAPM-FIN04] The accuracy of the method is dependent on the realism of the fault models. The fault types injected must be representative of real faults.
Scalability and computational No relevant gap or limitation has been identified.
Deployment [GAPM-FIN05] Tools always require some customization to be used in a given target system.
Learning curve [GAPM-FIN06] Requires specific knowledge on fault injection and a good knowledge and a detailed knowledge on the target system details.
Lack of automation The method is largely automatic.
Reference environment [GAPM-FIN07] It requires a prototype or a real system.
Costs [GAPM-FIN08] They could be moderate/high due to specialized knowledge required.

Standards No relevant gap or limitation has been identified. The method can be used in the context of the following standards: ISO 26262, IEC 62061, IEC TR 63074, ISO PAS 21448, ISO 13849, IEC 61508, ISO/IEC TR 24028:2020, ISO/IEC WD 23053
References
• [FIN1] R. Natella, D. Cotroneo, and H. Madeira, "Assessing Dependability with Software Fault Injection: A Survey", ACM Computing Surveys, Volume 48 Issue 3, February 2016. • [FIN2] F. Cerveira, R. Barbosa, H. Madeira and F. Araújo, "The Effects of Soft Errors and Mitigation Strategies for Virtualization Servers," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2020.2973146 • [FIN3] João A. Durães and Henrique S. Madeira "Emulation of Software Faults: A Field Data Study and a Practical Approach", IEEE Transactions on Software Engineering, vol. 32, no. 11, pp. 849-867, November 2006.

3.2 Simulation

This sub-group of methods describes the simulation-based V&V of selected properties. By the systematic development and exploitation of models, simulation-based approaches enable the automated execution of validation scenarios at early design phases of a project. Simulation focuses on the use of models that behave or operate like a given system to predict how the system would respond to defined inputs.

3.2.1 Simulation-Based Robot Verification

Name of the method: **Simulation-Based Robot Verification**
Short description Simulation-Based Robot Verification is proposed to assure a robots' safety. This method aims to decrease the cost of failures of robots before implementing them in real-world applications. At the same time, the method aims to prevent possible accidents and to avoid possible loss of life and properties by verifying the safety of systems.
Limitations
Functionality [GAPM-SBV01] Simulation based robot verification method only accepts files in STL and DAE formats as CAD data. In this case, many features of CAD data cannot be used in Simulation environments such as colouring, etc.
Accuracy [GAPM-SBV02] When a robot tested in a simulation environment is implemented in the real world, there may be some situations where simulation does not give the same results. The reason is that the physics engines of the simulations cannot meet the real world at 100 %. For this reason, when the

weights of the real-world model are entered, the error rate in the robot's controls increases and accuracy decreases. This situation may pose a risk for robotic applications which are critical for human life such as surgical robots.
Scalability and computational [GAPM-SBV03] High processor power and RAM are needed to run complex models in Simulation-based Fault injection. Since the processing load on the CPUs may increase in various simulation applications, a large amount of processing power may be needed to implement this method in complex simulation applications. This situation creates a limit in terms of computational and scalability. For instance, due to the complex simulation of body-in white system of OTOKAR's use case, this kind of problem might occur.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-SBV04] Method implementation requires the base knowledge about ROS, Gazebo, Python language.
Lack of automation No relevant gap or limitation has been identified.
Reference environment [GAPM-SBV07] The method is applied in simulation. It may require an adaptation to apply it in other environments.
Costs [GAPM-SBV05] When using this method, a large amount of hardware resources might be required depending on the number of tests to be performed.
Standards No relevant gap or limitation has been identified.
References
• [SBV1] Timperley, C. S., Afzal, A., Katz, D. S., Hernandez, J. M., & Le Goues, C. (2018, April). Crashing simulated planes is cheap: Can simulation detect robotics bugs early? In 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST) (pp. 331-342). IEEE.

3.2.2 Simulation-Based Testing for Human-Robot Collaboration

Name of the method: **Simulation-Based Testing for Human-Robot Collaboration**

Short description
Test-based simulation for human-robot collaboration provides the opportunity to evaluate the feasibility and performance of the system. Simulation allows evaluating particularly the layout or workplace planning, production reliability and, especially, the safety and efficiency of human-robot collaboration.
Limitations
Functionality [GAPM-SBT01] The method is missing an oracle of the Human-Robot Collaboration part that will provide real time diagnosis for the interaction between human and robots. This is especially important in the UC7: Human-Robot Collaboration in a Disassembly Process with Workers with Disabilities, where a diagnosis is required for the interaction between disabled humans and robots in a manufacturing and disassembly domain.
Accuracy [GAPM-SBT02] The accuracy of the simulation is a limitation as it can varies compared to real behaviour.
Scalability and computational [GAPM-SBT03] Simulation tools that use constraint-based modelling for assertion require much computational power and limit real-time applications.
Deployment No relevant gap or limitation has been identified.
Learning curve No relevant gap or limitation has been identified.
Lack of automation No relevant gap or limitation has been identified.
Reference environment [GAPM-SBT04] This method is applied in simulation.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified.
References

3.2.3 Test Optimization for Simulation-Based Testing of Automated Systems

Name of the method: **Test Optimization for Simulation-Based Testing of Automated Systems**
Short description The objective of test optimization is to cost-effectively test a system, i.e., reduce the cost of testing a system while the overall test quality is maintained. Test optimization could include test case selection, test case minimization, test case prioritization, etc. Test optimization could be also obtained using automatic test case generation: the process of generating test suites for a particular system.
Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy No relevant gap or limitation has been identified.
Scalability and computational [GAPM_TOS01] In order to apply the method to a new domain or scenario, empirical evaluation may be required. [GAPM_TOS02] In order to apply the method to a new domain or scenario, to gather enough historical data may be required.
Deployment [GAPM_TOS03] The method works mainly in Simulation environments. It may require an adaptation when the simulation tool used is changed.
Learning curve No relevant gap or limitation has been identified.
Lack of automation No relevant gap or limitation has been identified.
Reference environment [GAPM_TOS04] The method is applied in Simulation. It may require an adaptation to apply it in other environments, for example using the real robot.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified.
References

- [TOS1] A. Arrieta, S. Wang, G. Sagardui, L. Etxeberria. "Search-Based Test Case Prioritization for Simulation-Based Testing of Cyber-Physical System Product Lines" in Journal of Systems and Software. Volume 149, 2019, Pages 1-34, ISSN 0164-1212, https://doi.org/10.1016/j.jss.2018.09.055.
- [TOS2] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, L. Etxeberria. "Employing Multi-Objective Search to Enhance Reactive Test Case Generation and Prioritization for Testing Industrial Cyber-Physical Systems" in IEEE Transactions on Industrial Informatics, vol. 14, no. 3, pp. 1055-1066, March 2018, doi: 10.1109/TII.2017.2788019.
- [TOS3] Aitor Arrieta, Shuai Wang, Urtzi Markiegi, Ainhoa Arruabarrena, Leire Etxeberria, Goiuria Sagardui, Pareto efficient multi-objective black-box test case selection for simulation-based testing, Information and Software Technology, Volume 114, 2019, Pages 137-154, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2019.06.009.
- [TOS4] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui and L. Etxeberria, "Search-based test case generation for Cyber-Physical Systems," 2017 IEEE Congress on Evolutionary Computation (CEC), San Sebastian, 2017, pp. 688-697, doi: 10.1109/CEC.2017.7969377.
- [TOS5] Arrieta, A., Shuai Wang, Ainhoa Arruabarrena, Urtzi Markiegi, G. Mendieta and L. Etxeberria. "Multi-objective black-box test case selection for cost-effectively testing simulation models." Proceedings of the Genetic and Evolutionary Computation Conference (2018).

3.2.4 Virtual Architecture Development and Simulated Evaluation of Software Concepts

Name of the method: **Virtual Architecture Development and Simulated Evaluation of Software Concepts**
Short description The method deals with the efficient and reliable prototyping of complex systems involving cross-domain aspects by integrating heterogeneous components within holistic testing scenarios subject to goal-specific model fidelity and by systematically evaluating properties of interest in self-contained virtual runtime environments. It enables the automated generation, deployment, execution, and evaluation of test scenarios and test cases for early design verification and simulation of cross-domain systems with heterogeneous simulation models and network models.
Limitations
Functionality [GAPM-VAD01] For the application of the method in new use cases, connectors for additional communication protocols and simulation component types might need to be added. [GAPM-VAD02] For the design of virtual validation scenarios system-level model architecture has to be defined in detail prior to the implementation and adoption.
Accuracy

[GAPM-VAD03] The accuracy of virtual validation and simulation scenarios depends on the accuracy of the underlying simulation and behaviour model. There is a trade-off between accuracy on the one hand and simulation speed, resource consumption, and effort for constructing simulation models on the other hand.
Scalability and computational [GAPM-VAD04] The scalability of the validation approach is related to the maximum available time for executing simulation scenarios and single simulation steps. When real time components are connected, the real time represent the upper boundary for the execution of the simulation scenarios, method is scalable considering the trade-off between accuracy and performance of the simulation. [GAPM-VAD05] The simulation scenario can be deployed to multiple hosts to enable a distributed execution. The number of hosts is limited to 1000. The performance of the validation framework is influenced by the number of hosts due to the communication and synchronization overhead.
Deployment [GAPM-VAD06] The deployment of simulation components to host nodes requires the development or adaptation of platform-specific connectors, which depend on component type, communication protocols, and operating system. FERAL comprises a library of existing and available platform-specific connectors, which is continuously updated and extended. [GAPM-VAD07] Host node shall run Windows or Linux operating systems, preferably in 64-bit mode. [GAPM-VAD08] Host node must support Java version 11 or higher.
Learning curve [GAPM-VAD09] The learning curve depends on the concrete activities. The actual use and execution of simulation scenario is trivial and easy to learn. The construction of new simulation scenarios and the extension of existing scenarios is rather complex and requires some learning. The design of new simulation models and components requires the understanding of detailed method and tool insights, especially of the FERAL kernel and messaging paradigm.
Lack of automation [GAPM-VAD10] The implementation of the method in the FERAL tool framework is partially automated. Currently, the execution of simulation scenarios and the recording of data flows through the interfaces are fully automated. Further steps involved manual activities, such as the construction of simulation models, the definition and configuration of simulation scenarios, and the deployment of simulation components to host nodes. The evaluation of use-case specific properties can be automated by developing corresponding data processing and reporting engines.
Reference environment [GAPM-VAD11] Simulation is supported on pure model level, the called Model-in-the-Loop Test (MiL), on software level, the so-called Software-in-the-Loop Test (SiL), and for virtual hardware platforms, i.e. processor and network models to which the software components can be deployed in a virtual Hardware-in-the Loop (vHil) simulation.

Costs [GAPM-VAD12] For academic use and evaluation purposes, dedicated evaluation licenses are provided. For commercial use, customer-specific commercial licenses have to be purchased.
Standards [GAPM-VAD13] FERAL is applied in development projects for technical applications from different domains that consider specific process standards, such as IEC 61508 or ISO 26262. Nevertheless, FERAL is not qualified or operationally proven to be used in official release processes and audits with certification authorities.
References
• [MVV1] T. Kuhn, T. Forster, T. Braun, R. Gotzhein: Feral - framework for simulator coupling on requirements and architecture level. In: ACM/IEEE MEMOCODE, pp. 11–22, 2013 • [MVV2] P. Oliveira Antonino, J. Jahic, B. Kallweit, A. Morgenstern, T. Kuhn: Bridging the Gap between Architecture Specifications and Simulation Models. International Conference on Software Architecture (ICSA) Companion: 77-80, 2018 doi: http://dx.doi.org/10.1109/ICSA-C.2018.00029 • [MVV3] A. Bachorek, F. Schulte-Langforth, A. Witton, T. Kuhn, P. Oliveira Antonino: Towards a Virtual Continuous Integration Platform for Advanced Driving Assistance Systems. International Conference on Software Architecture (ICSA) Companion, 61-64 (2019), doi: http://dx.doi.org/10.1109/ICSA-C.2019.00018

3.2.5 Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance

Name of the method: **Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance**
Short description This is a method aimed at involving the end-user early in the validation process. Human factors, technology acceptance, and trust can be tested even before the system/robot is fully implemented by using virtual/augmented reality simulation and robot simulator [VUR1] [VUR2].
Limitations
Functionality [GAPM-VUR01] The simulation of the interaction between the end-user and the system/robot may be difficult to implement depending on the tasks to be performed and the level or realism desired.
Accuracy [GAPM-VUR02] This method relies on the accuracy of the system/robot simulation with which the end-user interacts and the realism of the human-robot interaction.

Scalability and computational [GAPM-VUR03] The method is designed to be used in a distributed environment in which each component can run in dedicated hardware. Different simulators can be combined in the same distributed simulation, so scalability will depend on the individual simulators used.
Deployment [GAPM-VUR04] Deployment can be complex if the method is planned to be used in the end-user facilities using augmented reality. Using virtual reality to recreate the facilities can also be complex and time consuming due to the need for creating realistic 3D of the facility.
Learning curve [GAPM-VUR05] It requires specific knowledge about networking and virtual reality/augmented reality development. In the application of the method used as reference, it requires knowledge about Unity3D and MQTT.
Lack of automation [GAPM-VUR06] This type of method requires participation of the end-users in order to obtain useful data.
Reference environment [GAPM-VUR07] This method is applied in a simulation environment, with the corresponding limitations (real-world representativeness, etc.)
Costs [GAPM-VUR08] The virtual/augmented reality simulator has to be implemented from scratch. Thus, cost may be high depending on the requirements.
Standards No relevant gap or limitation has been identified.
References
• [VUR1] Belmonte, L.; Garcia, A.S.; Segura, E.; Novais, P.J.; Morales, R.; Fernandez-Caballero, A. Virtual Reality Simulation of a Quadrotor to Monitor Dependent People at Home. IEEE Transactions on Emerging Topics in Computing, 2020. doi:10.1109/TETC.2020.30003. • [VUR2] Belmonte, L.M.; García, A.S.; Morales, R.; de la Vara, J.L.; López de la Rosa, F.; Fernández-Caballero, A. Feeling of Safety and Comfort towards a Socially Assistive Unmanned Aerial Vehicle That Monitors People in a Virtual Home. Sensors 2021, 21, 908. doi: 10.3390/s21030908.

3.2.6 V&V of Machine Learning-Based Systems Using Simulators

Name of the method: **V&V of Machine Learning-Based Systems Using Simulators**

Short description The traditional methods for V&V of a rule-based system are not effective for testing fuzzy machine learning-based models. Hence, for safety reasons these models are initially tested in simulators where the gap between the simulated environments and the real-world environments are being minimized.
Limitations
Functionality [GAPM-VVM01] Uncertainty in behaviour of machine learning-based systems. [GAPM-VVM02] Traditional rule-based methods not being efficient in testing Machine Learning (ML) systems.
Accuracy [GAPM-VVM03] ML algorithm's correct behaviour cannot be guaranteed by traditional software engineering approaches. Using simulators in V&V for ML will enable generation of annotated INFOR datasets. However, the accuracy limitations exist since data generated from sensor models in simulator does not represent data from real sensors. [VVM1] [GAPM-VVM04] Gap between test methods and evaluation criteria in real world and simulator and the test coverage of scenarios.
Scalability and computational [GAPM-VVM05] Coverage of scenario. Only few tests can be done on a real-world track. Using a simulator, we can test several instances of a model at the same time.
Deployment [GAPM-VVM06] Lack of industrial standard or systematic approach to integrate simulation into CI/CD pipeline.
Learning curve [GAPM-VVM07] Lack of knowledge about corner cases for test purposes. As Some preliminary knowledge is required to properly use a simulator [GAPM-VVM08] Lack of mature process for V&V of ML models. As Some preliminary knowledge is required of how to perform V&V for ML-based systems.
Lack of automation [GAPM-VVM09] Lack of automated tools for generating test cases and data of realistic sensor responses of real-world environment.
Reference environment No relevant gap or limitation has been identified.
Costs

[GAPM-VVM10] Modelling the real world in simulators requires manual intervention and is therefore time-consuming and expensive. [GAPM-VVM12] Real-world testing raises safety and ethical issues for test participants. [GAPM-VVM13] Open-source community working with these methods is limited and the quality of the software is not comparable with the commercial alternatives.
Standards [GAPM-VVM11] ISO26262 (functional safety) does not take ML into account while SOTIF (ISO 21448) is still in the development that comes with argumentations of ML based system to meet the safety requirements for critical applications.
References
• [VVM1] A. Ngo, M. P. Bauer, and M. Resch, "A Sensitivity Analysis Approach for Evaluating a Radar Simulation for Virtual Testing of Autonomous Driving Functions," *arXiv:2008.02725 [cs, eess]*, Oct. 2020, Accessed: Mar. 10, 2021. [Online]. Available: http://arxiv.org/abs/2008.02725.

3.3 Testing

This group of methods focuses on validating a system by executing it in the frame of so-called test cases. At least, a test case contains two fundamental sets of information: input data to be provided to the System Under Test (SUT), and a description of the expected output or behaviour. In order to execute a test case, an environment is used that allows to feed the SUT with the input data in a controlled manner, as well as to monitor its reactions.

3.3.1 Behaviour-Driven Model Development and Test-Driven Model Review

Name of the method: **Behaviour-Driven Model Development and Test-Driven Model Review**
Short description Uses automated model testing, test case generation and scenario review to ensure the correctness of behaviour models.
Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy [GAPM-MBT03] (Inherited from the used part-method Model-Based Testing, see section 3.3.6) The quality of the generated tests depends not only on the model and the tool, but also on the coverage criterion used to drive the generation of the tests. Tests generated to achieve control flow coverage reach a location in the system code where a problem could happen, but do not follow

through to a point where the problem would become observable on the outside interface. For this, data flow coverage or mutation coverage would be needed. For the integration in this method, this might lead to model behaviour that is not reviewed.
Scalability and computational [GAPM-MBT04] (Inherited from Model-Based Testing, see section 3.3.6) Conceptually, MBT uses enumerative or symbolic search over a state space. More complex systems have exponentially growing state spaces. Compared to, e.g., Model Checking, it does not necessarily need to cover the complete state space. Instead, it is sufficient to reach the requested coverage of the given model, but in the worst case, searching for this can take as long as full state space coverage. Various heuristics exist when to end the search but are rarely compared in detail. Method support to select a fitting heuristic based on the given model could mitigate this problem.
Deployment [GAPM-BHM01] Only limited and no integrated tool support for Behaviour-Driven Model Development and Test-Driven Model Review is available yet.
Learning curve If the tests used for model review shall also be used to test the implementation: [GAPM-MBT08] (Inherited from Model-Based Testing, see section 3.3.6) Behaviour models usable for MBT need to be (semi-)executable to generate tests. But to generate tests that reflect the requirements (what shall be done), there should be as little as possible how it is done – otherwise the generated tests would be more specific than the requirements and implementations under test might not pass the tests despite being perfectly in line with the requirements. This is hard to learn and get right. A method/guideline and/or tool support to find a reasonable balance of executability and abstraction might help. [GAPM-MBT09] (Inherited from Model-Based Testing, see section 3.3.6) Building behaviour models usable for MBT requires some additional knowledge and experience to correctly capture the test interface. A method/guideline and/or tool supporting the test interface definition might help. For the integration in this method, this is also important for the communication between modelling expert and domain expert.
Lack of automation [GAPM-BHM02] As of now, no recommender support for the modelling expert to better fit the given behaviours to be model is available.
Reference environment No relevant gap or limitation have been identified.
Costs [GAPM- BHM03] (Related to GAPM-MBT11 from the used part-method Model-Based Testing, see section 3.3.6):

Hardware Costs: Adding test case generation to modelling might need investment in dedicated test case generation equipment. [GAPM- BHM04] If the used test case generation approach produces more tests than necessary to cover all the functionality, review efforts might become infeasibly high. [GAPM- BHM05] Process change – investment: Replacing an established process is a considerable effort.
Standards The used base method Model-Based Testing is highly recommended for SIL3/4 by IEC 61508. No gaps related to standards known.
References

3.3.2 Assessment of Cybersecurity-Informed Safety

Name of the method: **Assessment of Cybersecurity-Informed Safety**
Short description: Black-box testing for security-informed safety of automated driving systems [ACS1]. To support black-box testing (that is testing without knowing the internal workings of the test object, see e.g. [ACS2]) as part of an independent evaluation, with the aim of producing an understanding of the interplay between safety and security, enabling a comparison of how well different ADSs can withstand safety-relevant security threats.
Limitations
Functionality [GAPM-ACS1] Development of an appropriate test suite for test facilities to assess cybersecurity, matching the feature class and sensor setup. [GAPM-ACS2] Develop and evaluating a coverage measure for a cybersecurity teste suite. [GAPM-ACS3] Capturing of post-attack behaviour and co-simulation with critical traffic scenarios to evaluate safety criteria need to be developed.
Accuracy [GAPM-ACS4] Validity, and a measure thereof, for co-simulation of post-attack behaviour with critical traffic scenarios to evaluate safety criteria need to be investigated.
Scalability and computational [GAPM-ACS5] This is an aspect that needs further investigation for the method that depends on the needed validity and successfulness of limiting the combinatorial state explosion in scenario-based testing.
Deployment

The goal of the investigation is limited to a small assessment test where a small test batch that can be shown to be representative for a whole class of ADS features, mainly to show feasibility and efficacy gains. [GAPM-ACS6] Integration with test track infrastructure is needed.
Learning curve [GAPM-ACS7] Requires high-level multi-disciplinary skill, the slope in the learning curve could be lowered by standardized information exchange formats and a more mature product under test.
Lack of automation [GAPM-ACS8] Potentially high level of automation but that also puts high requirements on the testing infrastructure, not present at this time.
Reference environment [GAPM-ACS9] The reference environment is real traffic where the vehicle is meant to operate, where the validity is hopefully preserved to proving grounds and by extension simulation. It is initially the proving ground vs simulation validity that is investigated here.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified. Relevant standards are fulfilled: ISO26262, ISO21434 and ISO21448.
References
• [ACS1] Skoglund, M. et al.: Black-Box Testing for Security-Informed Safety of Automated Driving Systems, VTS 2021-spring *(to appear)* • [ACS2] Forgács, István; Kovács, Attila (2019). Practical Test Design: Selection of Traditional and Automated Test Design Techniques.

3.3.3 Machine Learning Model Validation

Name of the method: **Machine Learning Model Validation**
Short description Model validation in machine learning automated systems serves to evaluate how a system performs and how safe it is when applied to input data other than the data used to train it.
Limitations
Functionality No relevant gap or limitation has been identified.

Accuracy [GAPM-MLV01] Machine learning is intrinsically a statistical approach. Therefore, the goal of machine learning techniques is generalizing well in most of the cases, accuracy on infrequent configurations could be limited. [GAPM-MLV02] Machine learning validation works well if the data are representative of the real situations that may happen in a real-world situation. If the available are limited or if they are not fully describing the reality, the accuracy will be lower.
Scalability and computational [GAPM-MLV03] If machine learning models are complex, also their application for validation purposes could require high computational resources.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-MLV04] Some knowledge about machine learning is required to properly interpret and improve the results.
Lack of automation Most validation or cross-validation approaches as well as the tuning of ML model parameters could be automated, so automation is usually not an issue.
Reference environment No relevant gap or limitation has been identified.
Costs No relevant gap or limitation has been identified since most machine learning tools are freely available.
Standards No relevant gap or limitation has been identified.
References

3.3.4 Model-Based Mutation Testing

Name of the method: **Model-Based Mutation Testing**
Short description Model-based mutation testing is a form of model-based testing. As coverage criterion to drive the test case generation, it uses mutations – artificial faults injected into the test model that the generated tests must be able to expose. The method shares the limitations of Model-Based Testing.

Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy [GAPM-MBT01] (Inherited from Model-Based Testing, see section 3.3.6) Level of model detail: as a black-box testing method, the method does not use any internal knowledge about the implementation, only its requirements and specification as inputs. Thereby, it can only systematically test what has been explicitly specified. It might therefore benefit from combinations with methods supporting model quality assurance or methods that learn models from operations data. [GAPM-MBT02] (Inherited from Model-Based Testing, see section 3.3.6) Factoring: Well-designed models strive to have each piece of information just in one place. The same cannot be guaranteed for an implementation, where bad habits like re-use by copy and paste as well as fundamental technical reasons can lead to having the same information in multiple locations. This might allow something to go wrong in one location and not in the other. Tests generated from a well-factored model might assume that having one test reaching a specific state (and potential fault) in the system under test is enough. In the implementation, this would then only help finding a problem at one of the locations, but not the other. [GAPM-MMT01] (Related to [GAPM-MBT03] from Model-Based Testing, see section 3.3.6) The quality of the generated tests depends not only on the model and the tool, but also on the coverage criterion used to drive the generation of the tests. In case of mutation testing, a badly selected set of mutation operators can limit the quality of the generated tests – both by stopping too early with a sub-optimal test suite and by not finding all interesting situations because the available effort is spent on less interesting situations provoked by too many mutants.
Scalability and computational [GAPM-MBT04] (Inherited from Model-Based Testing, see section 3.3.6) Conceptually, MBT uses enumerative or symbolic search over a state space. More complex systems have exponentially growing state spaces. Compared to, e.g., Model Checking, it does not necessarily need to cover the complete state space. Instead, it is sufficient to reach the requested coverage of the given model, but in the worst case, searching for this can take as long as full state space coverage. Various heuristics exist when to end the search but are rarely compared in detail. Method support to select a fitting heuristic based on the given model could mitigate this problem.
Deployment This method inherits the typical issues for integrating all Model-Based Testing approaches (See section 3.3.6) into a validation workflow: [GAPM-MBT05] Without an automated test execution environment, some benefits are limited. E.g. re-running multiple tests automatically to generate a failing short test that can be easily analysed, does not help if the of the originally failing long test needs to be stepped through manually anyway.

[GAPM-MBT06] Existing test execution environments might not fit the needs of tests from MBT (e.g. some things might not be observable in the test environment)
Learning curve [GAPM-MBT07] (Inherited from Model-Based Testing, see section 3.3.6) Building good test models needs a somewhat different skill set than building good tests. Not all good testers become also good test modelers. Interpretation of test results is not the same as with manual tests. The learning curve can be steep. [GAPM-MBT08] (Inherited from Model-Based Testing, see section 3.3.6) Behaviour models usable for MBT need to be (semi-)executable to generate tests. But to generate tests that reflect the requirements (what shall be done), there should be as little as possible how it is done – otherwise the generated tests would be more specific than the requirements and implementations under test might not pass the tests despite being perfectly in line with the requirements. This is hard to learn and get right. A method/guideline and/or tool support to find a reasonable balance of executability and abstraction might help. [GAPM-MBT09] (Inherited from Model-Based Testing, see section 3.3.6) Building behaviour models usable for MBT requires some additional knowledge and experience to correctly capture the test interface. A method/guideline and/or tool supporting the test interface definition might help.
Lack of automation GAPM-MBT10] (Inherited from Model-Based Testing, see section 3.3.6) Building the test model is not easily automatable.
Reference environment No relevant gap or limitation have been identified.
Costs [GAPM-MBT11] (Inherited from Model-Based Testing, see section 3.3.6) Hardware Costs: Shifting from personnel efforts to automation for test design for complex systems might need investment in dedicated test case generation and/or execution hardware equipment. [GAPM-MBT12] (Inherited from Model-Based Testing, see section 3.3.6) Human Resources: The effort of creating a test model is often seen as an otherwise unnecessary effort. It can be balanced with reduced test design efforts, but in situations where the benefits of repeated test case generation cannot be reaped for some reason, overall efforts might go up. Combining with methods for "model play-in" could reduce the problem. [GAPM-MBT13] (Inherited from Model-Based Testing, see section 3.3.6) Process change – investment: Replacing an established testing process with MBT is a considerable effort for a development/testing team.
Standards The base method Model-Based Testing is highly recommended for SIL3/4 by IEC 61508. No gaps related to standards known.

Potential gaps regarding certification, depending on tool implementations: [GAPM-MBT14] (Inherited from Model-Based Testing, see section 3.3.6) For certification, it is usually necessary to demonstrate why the test cases are there – therefore, an implementation of the method should provide sufficient traceability to link successful tests as evidence to the fulfilment of requirements. [GAPM-MBT15] (Inherited from Model-Based Testing, see section 3.3.6) For re-certification of new releases of a software, as little tests as possible should be changed, since all changed (new, modified and removed) tests would need to be re-evaluated in a review.
References

3.3.5 Model-Based Robustness Testing

Name of the method: **Model-Based Robustness Testing**
Short description This method uses a behaviour model of the system under test to derive unexpected inputs that can be used to check the implementation of the functionality for robustness.
Limitations
Functionality No general gaps known – different tools e.g. smart fuzzing tools have very diverse feature sets.
Accuracy No general gaps known – it depends on the exploration algorithms used by a specific implementation.
Scalability and computational [GAPM-MRT01] For complex systems, it is often just not feasible to run a robustness test suite of the size objectively needed.
Deployment No relevant gap or limitation has been identified.
Learning curve No relevant gap or limitation has been identified.
Lack of automation [GAPM-MRT02] Model-Based Robustness Testing feeds the system under test with unexpected input stimuli. There are almost endless possibilities to do so. A selection needs to be made which parts shall be tested for which unexpected inputs. If the selection is too big, the test will not terminate within a reasonable amount of time. The selection needs to be made manually, can get very detailed and needs sufficient experience.

Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-MRT03] The method depends on the availability of a behaviour model - if this needs to be built, personnel effort and costs go up.
Standards No relevant gap or limitation has been identified.
References

3.3.6 Model-Based Testing

Name of the method: **Model-Based Testing (MBT)**
Short description Model-Based Testing allows the derivation of tests from specification models and thereby automated test design. Specification models usually are behaviour models, but for some variants of the method can also be models of test scenarios, models of input and output interface and models of invariants.
Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy [GAPM-MBT01] Level of model detail: as a black-box testing method, the method does not use any internal knowledge about the implementation, only its requirements and specification as inputs. Thereby, it can only systematically test what has been explicitly specified. It might therefore benefit from combinations with methods supporting model quality assurance or methods that learn models from operations data. [GAPM-MBT02] Factoring: Well-designed models strive to have each piece of information just in one place. The same cannot be guaranteed for an implementation, where bad habits like re-use by copy and paste as well as fundamental technical reasons can lead to having the same information in multiple locations. This might allow something to go wrong in one location and not in the other. Tests generated from a well-factored model might assume that having one test reaching a specific state (and potential fault) in the system under test is enough. In the implementation, this would then only help finding a problem at one of the locations, but not the other. [GAPM-MBT03] The quality of the generated tests depends not only on the model and the tool, but also on the coverage criterion used to drive the generation of the tests. Tests generated to achieve control flow coverage reach a location in the system code where a problem could happen, but do not

follow through to a point where the problem would become observable on the outside interface. For this, data flow coverage or mutation coverage would be needed.
Scalability and computational [GAPM-MBT04] Conceptually, MBT uses enumerative or symbolic search over a state space. More complex systems have exponentially growing state spaces. Compared to, e.g., Model Checking, it does not necessarily need to cover the complete state space. Instead, it is sufficient to reach the requested coverage of the given model, but in the worst case, searching for this can take as long as full state space coverage. Various heuristics exist when to end the search but are rarely compared in detail. Method support to select a fitting heuristic based on the given model could mitigate this problem.
Deployment Typical issues for integrating MBT into a validation workflow are: [GAPM-MBT05] Without an automated test execution environment, some benefits are limited. E.g. re-running multiple tests automatically to generate a failing short test that can be easily analysed, does not help if the of the originally failing long test needs to be stepped through manually anyway. [GAPM-MBT06] Existing test execution environments might not fit the needs of tests from MBT (e.g. some things might not be observable in the test environment)
Learning curve [GAPM-MBT07] Building good test models needs a somewhat different skill set than building good tests. Not all good testers become also good test modelers. Interpretation of test results is not the same as with manual tests. The learning curve can be steep. [GAPM-MBT08] Behaviour models usable for MBT need to be (semi-)executable to generate tests. But to generate tests that reflect the requirements (what shall be done), there should be as little as possible how it is done – otherwise the generated tests would be more specific than the requirements and implementations under test might not pass the tests despite being perfectly in line with the requirements. This is hard to learn and get right. A method/guideline and/or tool support to find a reasonable balance of executability and abstraction might help. [GAPM-MBT09] Building behaviour models usable for MBT requires some additional knowledge and experience to correctly capture the test interface. A method/guideline and/or tool supporting the test interface definition might help.
Lack of automation [GAPM-MBT10] Building the test model is not easily automatable.
Reference environment No relevant gap or limitation have been identified.
Costs

[GAPM-MBT11] Hardware Costs: Shifting from personnel efforts to automation for test design for complex systems might need investment in dedicated test case generation and/or execution hardware equipment. [GAPM-MBT12] Human Resources: The effort of creating a test model is often seen as an otherwise unnecessary effort. It can be balanced with reduced test design efforts, but in situations where the benefits of repeated test case generation cannot be reaped for some reason, overall efforts might go up. Combining with methods for "model play-in" could reduce the problem. [GAPM-MBT13] Process change – investment: Replacing an established testing process with MBT is a considerable effort for a development/testing team.
Standards MBT is highly recommended for SIL3/4 by IEC 61508. No gaps related to standards known. Potential gaps regarding certification, depending on tool implementations: [GAPM-MBT14] For certification, it is usually necessary to demonstrate why the test cases are there – therefore, an implementation of the method should provide sufficient traceability to link successful tests as evidence to the fulfilment of requirements. [GAPM-MBT15] For re-certification of new releases of a software, as little tests as possible should be changed, since all changed (new, modified and removed) tests would need to be re-evaluated in a review.
References

3.3.7 Risk-Based Testing

Name of the method: **Risk-Based Testing**
Short description Risk-based testing uses identified risks in a system to prioritize and/or select test execution and sometimes even test development/generation.
Limitations
Functionality [GAPM-RBT01] Risk assessment artefacts and results need description in a format and approach that allows their usage by automated testing tools.
Accuracy [GAPM-RBT02] Prioritization by risk can reduce the remaining risk if test efforts need to be limited, but it cannot address the inherent incompleteness of testing. [GAPM-RBT03] Risk assessment artefacts and results need to be described on a technical / near implementation level in order to be usable for risk-based testing. Current concept level / preliminary architecture level assessments are difficult to utilize in testing.

Scalability and computational No relevant gap or limitation has been identified.
Deployment No relevant gap or limitation has been identified.
Learning curve No relevant gap or limitation has been identified.
Lack of automation [GAPM-RBT04] For security testing, risk-based testing is currently a manual activity, e.g. risk assessments results are considered for test planning, but there is no automated linkage from risk assessment to testing.
Reference environment No relevant gap or limitation have been identified.
Costs No relevant gap or limitation has been identified. Costs highly depend on concrete risk assessment and testing approaches.
Standards In certification of safety-critical systems, risk-based testing can usually not be used to reduce the sub-set of tests that is run. It can be used to prioritize test execution for regression tests.
References

3.3.8 Signal Analysis and Probing

Name of the method: **Signal Analysis and Probing**
Short description A method to validate signals on an IC based on using a tester setup that probes the IC to measure the signals on the chip. These signals are post-processed on the tester by means of complex signal analysis in order to assess the SoC's performance.
Limitations
Functionality [GAPM-SAP01] Without probing, the IC is a black box and intermediate processing stages cannot be evaluated. Errors are difficult to track down to the faulty subsystem in the IC.
Accuracy [GAPM-SAP02] Insufficient accuracy in testing due to missing intermediate test parameters.

Scalability and computational No relevant gap or limitation has been identified.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-SAP03] Interpretation of results at the end of the processing chain is more complex and requires a high level of skill to interpret them compared to analysis of intermediate results.
Lack of automation No relevant gap or limitation has been identified.
Reference environment No relevant gap or limitation have been identified.
Costs [GAPM-SAP04] Error tracking is slow because multiple test runs are required with different stimulation patterns to track down a problem.
Standards No relevant gap or limitation have been identified.
References

3.3.9 Software Component Testing

Name of the method: **Software Component Testing**
Short description Software Component Testing relies on tests on each SW component of the system under test in order to find poor and potentially incorrect program structures or failures. It is done at SW unit test and at SW units integration stages, which are iterations of tests involving the interfaces of each component to be put in the same system.
Limitations
Functionality [GAPM-SCT01] Current tool support is good but not prone to automation.
Accuracy No relevant gap or limitation has been identified.

Scalability and computational No relevant gap or limitation has been identified.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-SCT02] The method requires good software design and development skills.
Lack of automation [GAPM-SCT03] The method requires slight system design, and safety expertise, but the analysis results can be mostly automated and can be solidly gathered.
Reference environment [GAPM-SCT04] The method can be applied in any environment provided that the level of development of the component is completed.
Costs [GAPM-SCT05] Some tools are freely available as open-source software, some other requires quite an investment (thousands of euro) to be acquired.
Standards No relevant gap or limitations has been identified.
References

3.3.10 Test Parallelization and Automation

Name of the method: **Test Parallelization and Automation**
Short description: Complex systems require testing of a huge number of use cases and varieties of parameters. Expensive test equipment and time to market requirements demand efficient use of test resources and reliable result tracking.
Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy [GAPM-TPL01] With thousands of test cases, manual test execution can lead to forgotten tests. Computational test administration and scheduling ensures full coverage of defined test plans.

[GAPM-TPL02] Visual evaluation of test results makes it difficult to compare large number of test results and find correlations or corner cases between test parameters.
Scalability and computational [GAPM-TPL03] Manual administration of test resources does not optimize the utilization of the available test equipment like computational administration can do.
Deployment No relevant gap or limitation has been identified.
Learning curve No relevant gap or limitation has been identified.
Lack of automation [GAPM-TPL04] Manual administration of test resources (planning, execution, and result evaluation) is time consuming, error prone and leads to increased wear of the equipment.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-TPL05] Manual administration of test resources is time consuming and therefore more costly. [GAPM-TPL06] Expensive equipment is required for this method even if it is not used continuously 24h per day, causing waste of resources. [GAPM-TPL07] Time to market is increased cause by long test cycles.
Standards No relevant gap or limitation has been identified.
References

3.4 Runtime Verification

This group of methods focuses on verifying a system during execution. Today's automated systems are continuously growing in complexity, notably in what respects to the nature of their distributed architectures, the size and number of software components, and the amount of concurrency associated with these components. This makes most state-of-the art static verification techniques unscalable and impracticable. Runtime verification techniques are lightweight alternatives that make use of monitors, build based on formal specifications, that observe the target system and verify at execution time whether a set of specifications are met.

3.4.1 Dynamic Analysis of Concurrent Programs

Name of the method: **Dynamic Analysis of Concurrent Programs**
Short description Dynamic analysis of concurrent programs aims at coping with the inherent non-determinism in scheduling concurrent threads (processes or other units of computation) due to which it is difficult to catch rarely occurring but often critical bugs in synchronisation. To counter the problem, approaches such as extrapolating checkers, systematic exploration of all schedules up to some bound, and/or noise injection are used.
Limitations
Functionality [GAPM-DAC01] Approaches based on systematic exploration of schedules have problems with operations such as input/output, network communication, etc., which should be repeatedly executed with the same effect. Extrapolating checkers do not support all classes of bugs and programming constructions. Noise injection is often also optimised for specific classes of bugs and programming idioms. In general, there is much less support for multi-process programs than for multi-threaded ones. Approaches based on noise injection may be unusable for real-time systems.
Accuracy [GAPM-DAC02] Methods of dynamic analysis cannot offer formal guarantees that no bug of the given kind is missed. They can be used in less critical applications, during development phases, or in scenarios where more accurate approaches fail to scale enough or fail to cover all the needed programming constructions. Methods based on extrapolation may produce false alarms.
Scalability and computational [GAPM-DAC03] Approaches based on systematic exploration of schedules are in principle less scalable than extrapolation or noise injection, and they may fail to scale to truly large systems (though they may still be more scalable than heavy-weight formal verification approaches). Monitoring and analysing the run of the program under extrapolation-based analysis may also slow down the monitored program significantly (the base time can be multiplied from several times to even several thousand times). Noise-based injection may slow down the run of the monitored program comparably to extrapolation or even more. Despite that, successful applications have been reported even for programs of sizes up even millions of lines of code (though tens or hundreds of KLOCs are more common) [DAC1, DAC2].
Deployment [GAPM-DAC04] There exist solid tools for systematic exploration of schedules, extrapolation-based analysis, as well as noise injection. However, there are not too many of them, and quite some come from academia.
Learning curve

[GAPM-DAC05] The learning curve is not too steep: to start using techniques of dynamic analysis of concurrent programs, no truly special skills are needed. To master the techniques and to achieve the maximum possible efficiency, some experience in using them is, however, needed.

Lack of automation

[GAPM-DAC06] The methods may often be used fully automatically though to ensure the highest possible efficiency, they may require the user to suitably set their various parameters (e.g., type and strength of noise, etc.).

Reference environment

[GAPM-DAC07] The methods can be applied once some runnable version of the system to be analysed is available. A test harness is needed for full automation.

Costs

No relevant gap or limitation has been identified since the approach comes with moderate costs only.

Standards

[GAPM-DAC08] Not covered by standards.

References

- [DAC1] Rhodes, D., Flanagan, C., Freund, S.N.: BigFoot: Static Check Placement for Dynamic Race Detection. Proc. of PLDI'17, ACM, 2017.
- [DAC2] Fiedor, J., Muzikovska, M., Smrcka, A., Vasicek, O., Vojnar, T.: Advances in the ANaConDA Framework for Dynamic Analysis and Testing of Concurrent C/C++ Programs. Proc. of ISSTA'18, ACM, 2018.

3.4.2 Runtime Verification Based on Formal Specification

Name of the method: **Runtime Verification Based on Formal Specification**

Short description

This method consists in verifying the properties of a system by employing monitors that run alongside it. Such properties are usually impractical to verify offline (static formal verification) due to either complexity or to the very nature of the information to be verified (e.g., some data that is only known during runtime). Such monitors are to be generated automatically by being derived from formal specifications provided by the system developer.

Limitations

Functionality

[GAPM-RVF01] The correct functional behaviour of monitors is highly dependent on two mains aspects: 1) the quality of the data collected for monitors to consume and make decisions; 2) that monitors respect the scheduling policy defined for the system they are bound to observe and analyse. Moreover, both 1) and 2) are dependent on each other, in the sense that monitors not respecting

scheduling policies are not guaranteed to reason over the correct set of events, whereas processing wrong set of events may cause the monitors to take more time than expected to compute and thus break the pre-defined scheduling policy.
Accuracy [GAPM-RVF02] The accuracy of the verdicts given by the monitors is heavily dependent on user-defined specifications. Automating the generation of such monitors could drastically reduce human errors by abstracting the use of formal methods to verify a given user-defined specification's correctness. [GAPM-RVF03] The expressiveness of formal specification languages is notoriously limited. Combining multiple languages could partially mitigate the problem, but the accuracy of what needs to be specified could be compromised. An important point related to expressiveness is related to the limitation of specifying properties only on the observable part of the system or the ability to specify assumptions on the system behaviour. Current approaches to assumption-based runtime verification [RVF1] are limited to propositional models of the system.
Scalability and computational [GAPM-RVF04] Monitoring architectures imply an inevitable overhead on the target system. The monitoring architecture's actual impact on the target system's performance will depend on its complexity. The system's scalability could also be limited depending on the coupled monitors' complexity and the nature of the data they verify. [GAPM-RVF05] Local and distributed monitoring architectures will, most likely, have very different impacts on scalability and computational resources. [GAPM-RVF06] Monitors synthesized through hardware specification languages could be an option to mitigate the computational impact caused by software-based monitors. However, such an approach also entails additional expenses like, for instance, space, cost, power consumption, and weight.
Deployment [GAPM-RVF07] Guaranteeing that monitors do not affect the safety non-functional properties or, unintendedly, the target system's functional properties, is a serious concern for monitor architectures deployment. Formal methods could be applied to guarantee the compliance of the coupled monitoring architecture and the target system's safety aspects. [GAPM-RVF08] Deployment of monitors in systems not originally designed to be monitored/instrumented could require additional cost and effort because the designed system architecture is potentially not suitable to work alongside monitors at runtime. This is mainly due to the fact that no formal specifications of properties to be monitored were defined during the initial design stage. As a result, the definition of such formal specifications at later verification stages might require difficult rework at system's architectural level to accommodate the system instrumentation required by the monitors.
Learning curve

[GAPM-RVF09] In general, formal methods have a steep learning curve. However, the creation or the use of existing tools that can abstract part of the formalism could make the formal specification of the monitors a much easier task.

Lack of automation

[GAPM-RVF10] Although monitors do not need any human intervention once deployed, the instrumentation process itself is heavily human-dependent as there is not much research on automated system instrumentation.

Reference environment

No relevant gap or limitation have been identified.

Costs

[GAPM-RVF11] Costs of software-based monitors are mainly associated with research efforts, technology transfer from the academy to the industry, and software licensing. In the case of hardware-based monitors, there is an additional cost associated to the hardware itself and its instrumentation on the target system.

Standards

Standards are not a limitation, there are many standards that require the verification of safety conditions. For example, DO-178C, DO-278A, ISO26262 and ISO21448. The following forthcoming standards may also be of interest: IEEE P2846 and IEEE P7009.

References

- [RVF1] Cimatti, A., Tian, C., Tonetta, S.: "Assumption-Based Runtime Verification with Partial Observability and Resets." In Runtime Verification (RV 2019), pp. 165–184. Springer. doi: 10.1007/978-3-030-32079-9_10

3.4.3 Test Oracle Observation at Runtime

Name of the method: **Test Oracle Observation at Runtime**

Short description

Uses *Runtime verification based on formal specifications* to evaluate if and with which safety-margin the behaviour of a tested system is within the specification.

Limitations

Functionality

[GAPM-TOO01] The method only performs analyses on an individual behaviour, it is not exhaustive.

[GAPM-TOO02] Specification languages have certain expressiveness limits.

Accuracy

[GAPM-TOO03] The method needs to be combined with a manual or automated method to provide test stimuli, since it is only passively observing the behaviour.
Scalability and computational No relevant gap or limitation has been identified.
Deployment [GAPM-TOO04] To apply the method, the observed outputs need to be technically accessible in the test environment with the needed accuracy.
Learning curve [GAPM-TOO05] While less complex than other formal modelling notations, identifying what correct expected behaviour is and expressing it formally takes training.
Lack of automation [GAPM-TOO06] Formalisation of the specification could be supported.
Reference environment No relevant gap or limitation has been identified.
Costs No relevant gap or limitation has been identified since costs of the method are benign.
Standards No relevant gap or limitation has been identified.
References

3.5 Formal Verification

This group of methods aims to mathematically prove properties of a system or of information about it. We distinguish between formal verification for source code and formal verification in general.

3.5.1 Formal Source Code Verification

3.5.1.1 Deductive Verification

Name of the method: **Deductive Verification**
Short description Deductive verification is a method for verifying properties about a software system. Properties are usually expressed in some formal logic and then a series of mathematical rules/techniques are used to reason about these properties [DEV1, DEV2, DEV3, DEV4, DEV5, DEV6].

Limitations
Functionality [GAPM-DEV01] The functionality of deductive verification is dependent on the tools that implement this method. This typically involves the definition of a property called a pre-condition and another property called a post-condition. A series of deduction rules/steps are then applied to demonstrate that the postcondition can be derived from the precondition based on some intermediate steps that are specific to the program/system being modelled.
Accuracy [GAPM-DEV02] This method provides a proof of correctness, so the results are accurate and all possible valuations of the state space are reasoned about. However, the user usually reasons over an abstraction of the system, reasoning that a concrete implementation satisfies its abstract specification, so there can be a "reality gap" between the final, implemented system and the model that the proofs were carried out with respect to.
Scalability and computational [GAPM-DEV03] Deductive verification involves carrying out a series of proofs about a system. As systems increase in complexity, so does the associated proof effort which can present scalability issues depending on the prover used. Techniques such as formal refinement help to improve the scalability of this method but do not solve the problem completely.
Deployment [GAPM-DEV04] Some deductive verification approaches have been integrated well with existing programming paradigms such as VCC, OpenJML, and Why, which integrate well with programming languages like C and Java. However, some are standalone, such as Dafny, and offer little integration with other tools/formalisms.
Learning curve [GAPM-DEV05] Deductive verification usually requires expert knowledge of both the system to be verified as well as the tools to be used. Many of them offer a programmer-friendly environment with a mixture of automated and interactive theorem proving working in the background. However, the specification notation and the generated proofs are generally not familiar to software engineers and there is therefore a learning curve associated with both using the tools and interpreting the proof results. For example, some tools output a full proof derivation which can be difficult to parse for non-expert users.
Lack of automation [GAPM-DEV06] This approach has a high degree of automation as used by SMT solvers. However, some other tools facilitate both automatic and interactive proof (e.g., Event-B [DEV7]). This has the advantage of the user being able to contribute to a proof that the tool is struggling with but, in practice, interactive proofs can be quite time consuming.

Reference environment No relevant gap or limitation have been identified.
Costs [GAPM-DEV06] High costs associated with training expert users.
Standards No relevant gap or limitation has been identified since there are many standards that require the verification of safety conditions. For example, DO-178C, DO-278A, ISO26262 and ISO21448. The following forthcoming standards may also be of interest: IEEE P2846 and IEEE P7009.
References
• [DEV1] Filliâtre, J. Deductive software verification. Int J Software Tools Technology Transfer 13, 397 (2011). https://doi.org/10.1007/s10009-011-0211-0 • [DEV2] Hähnle R., Huisman M. (2019) Deductive Software Verification: From Pen-and-Paper Proofs to Industrial Tools. In: Computing and Software Science. LNCS, vol 10000. Springer, Cham. https://doi.org/10.1007/978-3-319-91908-9_1 • [DEV3] Towards deductive verification of control algorithms for autonomous marine vehicles S Foster, M Gleirscher, R Calinescu- arXiv preprint arXiv:2006.09233, 2020 - arxiv.org • [DEV4] Luo Z., Siegel S.F. (2018) Symbolic Execution and Deductive Verification Approaches to VerifyThis 2017 Challenges. In: ISoLA 2018. LNCS, vol 11245. Springer, Cham. https://doi.org/10.1007/978-3-030-03421-4_12 • [DEV5] Oortwijn W., Huisman M. (2019) Formal Verification of an Industrial Safety-Critical Traffic Tunnel Control System. In: Integrated Formal Methods. IFM 2019. Lecture Notes in Computer Science, vol 11918. Springer, Cham. https://doi.org/10.1007/978-3-030-34968-4_23 • [DEV6] Marieke Huisman, Rosemary Monahan, Peter Müller, Andrei Paskevich, Gidon Ernst. VerifyThis 2018: A Program Verification Competition. [Research Report] Université Paris-Saclay. 2019 • [DEV7] Abrial, Jean-Raymond. Modeling in Event-B: system and software engineering. Cambridge University Press, 2010.

3.5.1.2 Source Code Static Analysis

Name of the method: **Source Code Static Analysis**
Short description Static code analysis strives to analyse programs without executing them at all (i.e., purely on the syntactic level) or at least without executing them under the original semantics, meaning that some abstract semantics is used. There exist many different forms of static analysis based, e.g., on syntactic error patterns, data-flow analysis, extended type and effect analysis, abstract interpretation, or symbolic execution [SAN1, SAN2].

Limitations
Functionality Static analyses, when used for verification purposes, are often designed to look for a specific class of defects in a specific class of programs. [GAPM-SAN01] While there are many different analyses and tools for some classes of errors and programming constructions, there are classes of errors and programs for which not many (or no) analyses and tools are available or those that exist are of limited precision or scalability. Specific classes of errors and programming constructions not so well supported include dealing with low-level memory operations especially when dynamic linked data structures are used, dealing with concurrent programs, looking for defects related to non-functional properties such as performance, dealing with complex operations on arrays or strings, or dealing with various combinations of different data types (e.g., hash tables of linked lists of numeric values). [GAPM-SAN02] Even if some analysis exists for a given class of errors and programs, it often needs some fine-tuning for a given industrial setting so that its accuracy and scalability are appropriate.
Accuracy For different static analyses, the accuracy may vary very significantly. While there are sound static analyses guaranteeing conservative results (i.e., no error of a given kind is missed) implemented in certified tools, there are also analyses that are neither sound nor complete. [GAPM-SAN03] A significant risk of many of the sound approaches and tools is then that they may produce quite many false alarms, requiring manual checking, further fine-tuning of the analysis for the given code, and/or changing the code such that it passes the analysis. [GAPM-SAN04] Analyses not striving to be sound are typically designed such that they scale to huge code bases where they should be able to find at least some real errors while not reporting too many false alarms, but, of course, they may miss real errors.
Scalability and computational The scalability of static analyses does vary a lot: lightweight static analyses not required to be sound nor complete are capable of handling code bases having billions of lines of code while still producing useful results). [GAPM-SAN05] However, the more precise and conservative the analysis is required to be, the scalability typically decreases (though there are conservative analyses implemented in commercial certified tools such as AbsInt or Polyspace that are in routine use for checking specific classes of properties of real-life critical industrial code, e.g., in the area of automotive or aerospace industries).
Deployment A basic application of a static analyser supporting some classes of errors and programs may be easy when not insisting too much on accuracy and scalability. [GAPM-SAN06] Otherwise, much more effort is needed in the deployment: Fine-tuning existing analysers for a given industrial setting requires significant expertise and ideally a dedicated verification engineer (or verification group, depending on the extent of the verification tasks). It may also be needed to change the coding style used by the developers in order to facilitate scalability and accuracy of static analyses on the produced code. Developing a new static analysis in case an

appropriate one is missing will typically require a very high level of expertise. Moreover, while some static analysers may easily integrate with the development tools used, others may require dedicated code be developed to facilitate the integration.

Learning curve

[GAPM-SAN07] As indicated already above, basic usage of existing static analysers does not require any special skills. However, fine-tuning the analysers to be efficient and accurate in a given setting requires significant expertise (and typically a specialised verification engineer). If a suitable analysis is missing completely, a very high level of specialist education may be required.

Lack of automation

[GAPM-SAN08] While a basic application of a static analyser may be fully automated, serious usage in a larger industrial setting will typically require some fine-tuning.

Reference environment

[GAPM-SAN09] Some static analyses may be applicable on code fragments that are not even runnable (the fact they pass syntax analysis may be sufficient). However, other static analyses may require code that is runnable and some even code with a test harness.

Costs

Basic usage of static analysis may come with rather moderate costs.

[GAPM-SAN10] However, more serious applications may incur significant costs. Indeed, while some static analysis tools may come for free (including some commercial tools when used in the open-source domain), the costs of using some of the current, commercially available tools may reach high tens of thousands of EUR per year when certification, customisation, and support are needed. Further costs may then be associated with a need to have a dedicated verification engineer (or engineers), which can, however, replace some number of testers.

Standards

[GAPM-SAN11] Some static analysers support standards such as MISRA C/C++, ISO 26262, DO-178B/C, IEC 61508, but many of them do not address any standards.

References

- [SAN1] Krena, B., Vojnar, T.: Automated Formal Analysis and Verification: An Overview. International Journal of General Systems, 42(4), 2013.
- [SAN2] Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis, Springer-Verlag, 2005.

3.5.2 General Formal Verification

3.5.2.1 Behaviour-Driven Formal Model Development

Name of the method: **Behaviour-Driven Formal Model Development**
Short description This approach uses scenarios to specify example use cases of the system in question. Based on a predefined set of requirements and these scenarios, a formal model of the system is devised along with the desired verification conditions.
Limitations
Functionality [GAPM-BFM01] This is usually more suitable to discrete than it is to continuous systems.
Accuracy [GAPM-BFM02] This method requires input from both formal methods and domain experts, so its accuracy depends on the quality of the communication and understanding between them.
Scalability and computational [GAPM-BFM03] The limitations of this method are those which are present in the tool support that is used to develop the formal model and the expressivity of the formalism chosen to define the verification conditions.
Deployment No relevant gap or limitation has been identified since many tools can facilitate the development of formal models.
Learning curve No relevant gap or limitation has been identified. This approach has the advantage of splitting the verification task between different experts so that one does not need to be fully trained on the other's topic. It thereby addresses a gap of classical formal methods e.g. Model Checking (see [GAPM-MCH05])
Lack of automation [GAPM-BFM04] This is an iterative process that requires frequent communication between individuals.
Reference environment No relevant gap or limitation have been identified.
Costs No relevant gap or limitation has been identified.

Standards No relevant gap or limitation has been identified since standards are not a limitation, there are many standards that that require the verification of safety conditions. For example, DO-178C, DO-278A, ISO26262 and ISO21448. The following forthcoming standards may also be of interest: IEEE P2846 and IEEE P7009.
References

3.5.2.2 Formal Requirements Validation

Name of the method: **Formal Requirements Validation**
Short description Requirements are formalized in a formal language, typically a temporal logic, and analysed to find defects in the specification such as inconsistencies, incompleteness, errors, or unrealizability.
Limitations
Functionality [GAPM-FRV01] Since the requirements specification is validated against the intentions of the requirements engineer and the needs of the stakeholders, the quest for new checks to perform to find new issues is always open.
Accuracy [GAPM-FRV02] There is always a gap between the natural language text and its formal counterpart. This limits the accuracy of the formal analysis with respect to the informal specification. It often happens that issues found in the formal properties are due to missing assumption or wrong choice in the formalization step.
Scalability and computational [GAPM-FRV03] Since the underlying formal problem is usually a problem of satisfiability or realizability for temporal logics such as LTL, one of the main issues is the scalability of the procedures. In some cases, considering for example first-order logic, the problem may be even undecidable.
Deployment [GAPM-FRV04] Some formal requirements tools, such as NASA's Formal Requirements Elicitation Tool (FRET) [FRV9], support the generation of verification conditions (in CoCoSim) based on the formalised requirements [FRV10]. However, there is often a gap between the requirements themselves and their associated verification conditions due to the requirements being expressed at a higher level of abstraction than the implementation.
Learning curve [GAPM-FRV05] The formalization of requirements is still a manual process and requires that the domain engineer, expert in the domain of the requirements, learns the formal language.

Lack of automation [GAPM-FRV06] The formalization of natural language requirements remains largely a manual step.
Reference environment No relevant gap or limitation have been identified.
Costs [GAPM-FRV07] The human effort in the formalization of requirements and in the analysis of the formal results is a limitation for a more widespread industrial adoption.
Standards No relevant gap or limitation has been identified since there are many standards that require the verification of safety conditions. For example, DO-178C, DO-278A, ISO26262 and ISO21448. The following forthcoming standards may also be of interest: IEEE P2846 and IEEE P7009.
References

- [FRV1] Ingo Pill, Simone Semprini, Roberto Cavada, Marco Roveri, Roderick Bloem, Alessandro Cimatti: Formal analysis of hardware requirements. DAC 2006: 821-826
- [FRV2] Roderick Bloem, Roberto Cavada, Ingo Pill, Marco Roveri, Andrei Tchaltsev: RAT: A Tool for the Formal Analysis of Requirements. CAV 2007: 263-267
- [FRV3] Alessandro Cimatti, Marco Roveri, Stefano Tonetta: Requirements Validation for Hybrid Systems. CAV 2009: 188-203 Requirements Validation for Hybrid Systems. CAV 2009: 188-203
- [FRV4] Alessandro Cimatti, Marco Roveri, Angelo Susi, Stefano Tonetta: Validation of requirements for hybrid systems: A formal approach. ACM Trans. Softw. Eng. Methodol. 21(4): 22:1-22:34 (2012)
- [FRV5] Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, Stefano Tonetta: SMT-based satisfiability of first-order LTL with event freezing functions and metric operators. Inf. Comput. 272: 104502 (2020)
- [FRV6] Amir Pnueli, Roni Rosner: On the Synthesis of a Reactive Module. POPL 1989: 179-190
- [FRV7] Dana Fisman, Orna Kupferman, Sarai Sheinvald-Faragy, Moshe Y. Vardi: A Framework for Inherent Vacuity. Haifa Verification Conference 2008: 7-22
- [FRV8] Koen Claessen: A Coverage Analysis for Safety Property Lists. FMCAD 2007: 139-145
- [FRV9] Giannakopoulou, D., Mavridou, A., Rhein, J., Pressburger, T., Schumann, J. and Shi, N., 2020. Formal requirements elicitation with fret.
- [FRV10] Mavridou, Anastasia, Hamza Bourbouh, Pierre-Loic Garoche, and Mohammad Hejase. Evaluation of the FRET and CoCoSim tools on the ten Lockheed Martin cyber-physical challenge problems. NASA, Tech. Rep., oct (2019).

3.5.2.3 Model Checking

Name of the method: **Model Checking**
Short description Given a formal model of the system and a formal specification of the properties, model checking provides automated procedures to prove or disprove that the system satisfies the property. This is achieved via an exhaustive examination of the state space.
Limitations
Functionality [GAPM-MCH01] When the property is satisfied, most model checkers do not provide any further information. Certifying model checking [MCH1] is an approach that provides also a deductive proof extracted from the model checker internals. Only few cases extended the approach beyond invariant properties [MCH2].
Accuracy [GAPM-MCH02] The results are accurate because every system state is explored in the process. However, model checkers usually require an abstract model of the system that is written in a different language to the final implementation. As a result, there can be a slight mismatch between the model that has been checked and the final, implemented solution.
Scalability and computational [GAPM-MCH03] The method exhaustively explores the state space of the system, which is often exponential in the number of variables and the number of system components executing in parallel. This problem is known as the state space explosion, which can make the application of model checking to industrial use cases impractical. In order to tackle this problem one has to choose suitable reduction and abstraction techniques, which basically consist in reducing the number of state transition paths to be explored by avoiding visiting those unnecessary paths that will not affect the verification outcome and limiting the number of system variables to a minimum by abstracting away those details that are not relevant to the system properties to be verified. [GAPM-MCH04] In case of hybrid systems or software systems, the model checking problem is usually undecidable. Although automated incomplete procedures exist, they may not scale up to the complexity required in an industrial context.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-MCH05] System and properties must be formalized into a formal language. Thus, it is required to have some background in formal methods such as temporal logic and associated proof strategies.
Lack of automation

No relevant gap or limitation has been identified since once the model and properties to be checked are formalized, then the process of checking the model against the properties is automatic.
Reference environment No relevant gap or limitation have been identified.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified.
References
• [MCH1] Kedar S. Namjoshi: Certifying Model Checkers. CAV 2001: 2-13 • [MCH2], Marco Roveri, Stefano Tonetta: Certifying Proofs for LTL Model Checking. FMCAD 2018: 1-9, https://dblp.org/pid/19/3686.html

3.5.2.4 Reachability-Analysis-Based Verification for Safety-Critical Hybrid Systems

Name of the method: **Reachability-analysis-based verification for safety-critical hybrid systems**
Short description: Reachability analysis constitutes a powerful –yet not mature enough– verification method for validating safety and robustness of safety-critical cyber-physical systems. It is based on a combination of formal methods and applied mathematics using techniques such as zonotopes, in order to co-analyse the continuous dynamics of (linear or non-linear) physical processes controlled by a discrete controller and verify that the control logic will reside only in acceptable states.
Limitations
Functionality [GAPM-RAV01] A major limitation with regard to functionality is that (by itself) this approach can only typically provide results for a finite time horizon of evolution of system behaviour. Invariant extraction and satisfaction checking can be used to alleviate this situation, but this is not always possible and even when it is, identification of the invariant itself can be quite computationally expensive.
Accuracy [GAPM-RAV02] The verification of safety and robustness of safety-critical systems has to fulfil several accuracy requirements especially in cases when random faults are present during the controller's operation. Reachability analysis in cases of those control perturbations require an over approximation of reachable (numerical) states that lead to state space explosion problems, setting the analysis incomplete.

Scalability and computational [GAPM-RAV03] Especially for non-linear dynamical and stiff systems, V&V methods have to cope with large system complexity, which usually form a hard problem to solve. Reachability analysis is not scalable enough in its current state as more efficient and approximation-sound techniques should be developed to tackle the problem.
Deployment [GAPM-RAV04] Reachability analysis for hybrid system verification usually requires an advanced expertise from the system engineer to be deployed (knowledge of hybrid automata, linearization and system hybridization). Currently it lacks fully automated and integrated characteristics as the engineer needs to specify its system in different representation semantics bounded by each reachability analysis solution.
Learning curve [GAPM-RAV05] Engineers need to have multidisciplinary capabilities in applied mathematics, control design and formal verification (reachability analysis principles).
Lack of automation [GAPM-RAV06] Semi-automated method lacking full integration with current model-based design frameworks.
Reference environment [GAPM-RAV07] Currently reachability analysis is at TRL3-4 and is being used in an ad-hoc manner during control system design and validation phases.
Costs [GAPM-RAV08] Large human resources demand cost
Standards [GAPM-RAV09] No standard currently addressed.
References

3.5.2.5 Theorem Proving and SMT Solving

Name of the method: **Theorem proving and SMT solving**
Short description Usually, theorem provers and SMT solvers are used as back-end proof support in deductive verification systems and model checkers where the user describes properties to be verified using a formal logic. Typically, these properties focus on safety and liveness aspects of the system. This method produces a proof of correctness for a system that it obeys the specified properties [TPS5, TPS6].

Limitations
Functionality [GAPM-TPS01] Tools for theorem proving and SMT solving usually focus on using a single logic to represent formal properties [TPS1, TPS2, TPS3, TPS4]. Incorporating multiple logical representations could potentially be beneficial.
Accuracy [GAPM-TPS02] This method provides a proof of correctness, so the results are accurate and all possible valuations of the state space are reasoned about. However, the user usually reasons over an abstraction of the system so there can be a "reality gap".
Scalability and computational [GAPM-TPS03] This technique involves carrying out a series of proofs about a system. As systems increase in complexity, so does the associated proof effort which can present scalability issues depending on the prover used. Techniques such as formal refinement help to improve the scalability of this method but do not solve the problem completely.
Deployment No relevant gap or limitation has been identified.
Learning curve [GAPM-TPS04] The supporting tools can be either automatic or interactive. In either case, the properties/system must be specified in a given logic and this can often require expert knowledge of the tools. [GAPM-TPS05] In the interactive case, since the user must contribute to the proof itself, the user must have a deep understanding of how the tools work.
Lack of automation [GAPM-TPS06] These tools tend to have a high degree of automation but when a proof attempt fails then the user may need to rewrite parts of their system specification and properties in order for the proof to be discharged.
Reference environment No relevant gap or limitation have been identified.
Costs [GAPM-TPS07] High costs associated with training expert users.
Standards

Standards are not a limitation, there are many standards that that require the verification of safety conditions. For example, DO-178C, DO-278A, ISO26262 and ISO21448. The following forthcoming standards may also be of interest: IEEE P2846 and IEEE P7009.
References
• [TPS1] KeYmaera: a hybrid theorem prover for hybrid systems (system description) A. Platzer and J.-D. Quesel, Automated Reasoning, Springer, Berlin (2008), pp. 171-178 • [TPS2] Z. Manna, N. Bjoerner, A. Browne, and E. Chang, STeP: The Stanford Temporal Prover, LNCS, Vol. 915, 1995, pp. 793–794. • [TPS3] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. volume 10510 of LNCS, pages 50–66. Springer, 2017. • [TPS4] Deductive verification of hybrid systems using step, Z. Manna and H.B. Sipma Hybrid Systems: Computation and Control, Springer, Berlin (1998), pp. 305-318 • [TPS5] P. Bagade, A. Banerjee, S.K.S. Gupta, Chapter 12 - Validation, Verification, and Formal Methods for Cyber-Physical Systems, In Intelligent Data-Centric Systems, Cyber-Physical Systems, Academic Press, 2017 • [TPS6] A Survey on Theorem Provers in Formal Methods M. Saqib Nawaz, Moin Malik, Yi Li, Meng Sun and M. Ikram Ullah Lali, https://arxiv.org/pdf/1912.03028.pdf

3.6 Semi-Formal Analysis

This group of methods deals with system evaluation by using structured means whose application does not result in a mathematical proof. The methods have been divided into two sub-groups: one for SCP-focused semi-formal analysis and another for general semi-formal analysis.

3.6.1 SCP-Focused Semi-Formal Analysis

3.6.1.1 Human Interaction Safety Analysis

Name of the method: **Human interaction safety analysis (HISA)**
Short description Safety analysis method for human interaction with automated systems. The aim is to reduce interaction risks and provide evidence to a safety case [HIS1].
Limitations
Functionality [GAPM-HIS01] The method is a process for systematic analysis of interactions between a human and automated system. Lacking functionality is that the process currently only considers a specific type of interactions and need to be elaborated to become more generic. [GAPM-HIS02] It also does not currently take the impact of cybersecurity into account.
Accuracy

[GAPM-HIS03] Risk assessment is based on qualitative expert judgment. Quantitative metrics could be added where applicable.
Scalability and computational [GAPM-HIS04] V&V of analysis results require real-world user tests which can be resource heavy and lack scalability, and potentially be dangerous as humans need to be involved. An open question is if simulation can be used to lower V&V time and costs.
Deployment [GAPM-HIS05] Current lack of tool support for efficient use of the method. [GAPM-HIS06] Also lack integration of verification tools (e.g. simulation) to back up claims from expert judgment.
Learning curve [GAPM-HIS07] Requires highly skilled personnel as it relies on both human factors and functional safety expertise.
Lack of automation [GAPM-HIS08] Currently there is no custom tool for this method.
Reference environment No relevant gap or limitation has been identified.
Costs No relevant gap or limitation has been identified.
Standards No standard specifically references this method as it is not an established method, however, it could become useful as part of an argumentation for fulfilling ISO PAS 21448, and potentially functional safety standards such as ISO 26262.
References
• [HIS1] Warg, F., Ursing, S., Kaalhus, M. and Wiik, R., 2020, January. Towards Safety Analysis of Interactions Between Human Users and Automated Driving Systems. In 10th European Congress on Embedded Real Time Software and Systems (ERTS 2020).

3.6.1.2 Intrusion Detection for WSN based on WPM State Estimation

Name: **Intrusion detection for wireless sensor networks based on WPM state estimation**
Short description Intrusion detection system which targets WSN hardware platforms and help identify incoming attacks by means of (node) state estimation via the Weak Process Modelling of the attacks.
Limitations

Functionality [GAPM-IDS01] The method helps identifying attacks but does not offer support to WSN operators for performing forms of active self-defence (i.e., there is no active reactions to attacks)
Accuracy Accuracy is not a limitation of the method itself, but it depends only on how the attack is modelled.
Scalability and computational No relevant gap or limitation has been identified.
Deployment No relevant gap or limitation has been identified.
Learning curve No relevant gap or limitation has been identified.
Lack of automation [GAPM-IDS02] Attack families have to be modelled in a WPM to allow the IDS to detect them. The automation of WPM creation is possible but unfeasible in most situations since it requires separate WPMs for each combination of attack vs. application running on the node.
Reference environment [GAPM-IDS03] It can be applied to in-lab experiments; not yet tested in a real-world scenario.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified.
References

3.6.1.3 Kalman Filter-Based Fault Detector

Name: **Kalman filter-based fault detector**
Short description: Use of historical data collected from a system via sensors to build mathematical models to be used to detect system's faults with respect to its nominal operation.
Limitations
Functionality [GAPM-KFB01] For model-based approaches, conditions for Identification with respect to detection are much more conservative. [GAPM-KFB02] Lack of rigorous mathematical guarantees about whether a fault can be detected.

Accuracy [GAPM-KFB03] The method provides good accuracy in the context of systems that exhibit a linear behaviour. Model accuracy can be limited if the system's dynamics are more complex.
Scalability and computational [GAPM-KFB04] Although the method itself is theoretically scalable, there are scalability and computational issues when applications to complex systems are considered. The main issue is related to the coding part due to the lack of powerful tools.
Deployment [GAPM-KFB05] There is lack of tools that can be directly used for the real-world applications.
Learning curve [GAPM-KFB06] Due to lack of tools, theoretical and programming skills are required.
Lack of automation [GAPM-KFB07] The code needs to be adapted case by case.
Reference environment No relevant gap or limitation has been identified.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation have been identified.
References

3.6.1.4 Model-Based Safety Analysis

Name: **Model-Based Safety Analysis**
Short description Model-Based Safety Analysis (MBSA) is an approach in which the system and safety engineers share a common system model created using a model-based development process. By extending the system model with a fault model as well as relevant portions of the physical system to be controlled, automated support can be provided for safety analysis, in particular for FMEA and FT generation. *Failure Logic Analysis (FLA)* and *Model-Based Fault Injection (MBFI) for safety analysis* are building blocks for Model-Based Safety Analysis that can be applied for V&V at different levels across the project life cycle.
Limitations

Functionality [GAPM-MSA01] Use of results in relation with other V&V methodologies may be explored (e.g. optimization of failure injection tests). [GAPM-MSA02] The FLA and MBFI building blocks may be connected, for their application in the same scenarios at different levels consistently. [GAPM-MSA03] Extensions in support of cybersecurity for FLA may be explored in real scenarios.
Accuracy [GAPM-MSA04] A potential gap between the conceptual level analysis and the real system could arise due to the fact that analysis requires to abstract away the critical aspects about both the system under development and the external environment.
Scalability and computational [GAPM-MSA05] The Failure propagation algorithm for FLA may be improved for large systems, impacting FMEA and FT generation. [GAPM-MSA06] The automated analysis for MBFI may be subject to the state-explosion problem, impacting the effectiveness of verification.
Deployment The method is well-supported by tools (e.g. CHESS, CHESS-FLA, xSAP, ocra, COMPASS). [GAPM-MSA07] Deployment for the use cases has to be experimented.
Learning curve [GAPM-MSA08] The method requires model-based design skills, as well as safety expertise.
Lack of automation The method requires system design and some safety expertise for failure definition; however, the analysis results are mostly automated. [GAPM-MSA09] Improvements for FMEA and FT generation for FLA are required.
Reference environment No relevant gap or limitation has been identified. The method may be applied at development level in TRL6-7 environments and prototypes.
Costs No relevant gap or limitation has been identified. No hardware and software specific costs, as the building blocks are available with the CHESS open-source tool. In particular, the xSAP and OCRA tools are freely available for non-commercial applications. The COMPASS tool is freely available for ESA member states.
Standards

No relevant gap or limitation has been identified. The method is conceived to fulfil safety standards (e.g. ECSS, EN 50129, SAE-ARP-4754, SAE-ARP-476), and may be applied in VALU3S for the medical and agriculture domains (e.g. by applying the CEI EN 62304, ISO 14971, IEC 61508, ISO 26262 standards).

References

- [MSA1] Wallace, Modular Architectural Representation and Analysis of Fault Propagation and Transformation, in proceedings of 2nd International Workshop on Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA 2005).
- [MSA2] R. F. Paige, L. M. Rose, X. Ge, D. S. Kolovos, and P. J. Brooke. FPTC: automated safety analysis for domain-specific languages. In Models in Software Engineering, M. R. Chaudron (Ed.). Lecture Notes In Computer Science, Vol. 5421. Springer-Verlag, Berlin, Heidelberg, pp. 229-242, 2009Space Product Assurance: Software product assurance, id. ECSS-Q-ST-80 issue C, 06.03.2009.
- [MSA3] B. Gallina and S. Punnekkat, "FI4FA: A Formalism for Incompletion, Inconsistency, Interference and Impermanence Failures Analysis," in Proc. of EUROMICRO, ser. SEAA '11. IEEE Computer Society, 2011, pp. 493–500.
- [MSA4] B. Gallina, M. A. Javed, F. U. Muram, and S. Punnekkat, "Model-driven dependability analysis method for component-based architectures," in Euromicro-SEAA Conference. IEEE Computer Society, 2012.
- [MSA5] Gallina, B., Sefer, E., and Refsdal, A. (2014). Towards safety risk assessment of socio-technical systems via failure logic analysis. In Proceedings of the 2014 IEEE International Symposium on Software Reliability Engineering Workshops, pages 287–292.
- [MSA6] B. Gallina and Z. Haider, A. Carlsson, S. Mazzini S. Puri, "Multi-concern Dependability-centered Assurance for Space Systems via ConcertoFLA", International Conference on Reliable Software Technologies- Ada-Europe 2018, Lisbon, June 2018
- [MSA7] Mazzini S., J. Favaro, S. Puri, L. Baracchi., "CHESS: an open source methodology and toolset for the development of critical systems", 2nd International Workshop on Open Source Software for Model Driven Engineering (OSS4MDE), Saint-Malo, October 2016.
- [MSA8] CHESS Dependability Guide – FLA (https://www.eclipse.org/chess/publis/CHESS_DependabilityGuide.pdf)
- [MSA9] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli and G. Zampedri. The xSAP Safety Analysis Platform. In Proceedings of *TACAS 2016*. Eindhoven, The Netherlands, April 2-8, 2016.
- [MSA10] M.Bozzano, A.Cimatti, J.-P.Katoen, V. Y.Nguyen, T.Noll and M.Roveri. Safety, Dependability, and Performance Analysis of Extended AADL Models. *The Computer Journal, 54(5):754-775, 2011.*
- [MSA11] M. Bozzano, A. Cimatti, J.-P. Katoen, P. Katsaros, K. Mokos, V.Y. Nguyen , T. Noll, B. Postma and M. Roveri. Spacecraft Early Design Validation using Formal Methods. *Reliability Engineering & System Safety 132:20-35*. December 2014.

3.6.1.5 Model-Based Threat Analysis

Name: **Model-Based Threat Analysis**
Short description Automated threat analysis based on a. a system model, enhanced with security properties and security related information. b. a threat model, containing formalized vulnerabilities, attacks and weaknesses. Results are potential risks.
Limitations
Functionality [GAPM-MTA01] The method itself is complete, but better integration into model-based engineering could enhance functionality and work is ongoing regarding an interconnection with SysML.
Accuracy [GAPM-MTA06] The accuracy of the approach is not limited by the method itself, but limited by the level of detail available in the system model and the completeness of the threat model.
Scalability and computational [GAPM-MTA02] System models can be created on different granularities, but threat and risk results from different granularities are not connected. We aim towards an integration of threats from "lower level" models into "higher level" models.
Deployment [GAPM-MTA03] The tool support is integrated in Enterprise Architect but integration with other related tools could enhance usability and offer features like access control, versioning and auditing.
Learning curve [GAPM-MTA04] Application of tools is easy, but creation and maintenance of threat models requires security expert knowledge.
Lack of automation [GAPM-MTA05] Currently the impact (risk = impact * likelihood) can be automatically identified based on assets. Likelihood needs expert judgment. In the future the likelihood should be also automatically identified based on attack potential.
Reference environment No relevant gap or limitation has been identified.
Costs No relevant gap or limitation has been identified.
Standards

[GAPM-MTA07] Tool (ThreatGet) was originally developed for the automotive domain and fully supports ISO/SAE 21434. We aim to enhance support to other domains and their respective security standards (IoT, industrial with IEC 62443).
References

3.6.1.6 Risk Analysis

Name: **Risk Analysis**
Short description Quantitative deterministic Risk Analysis methodology to evaluate the risk level of a system considering a wide set of threat scenarios (attacks and incidents), as well as the effectiveness of proper countermeasures. This methodology can be tailored on the specific features of a system, such as different components or different threats to be examined within the risk analysis process. In fact, starting from the modelling of the system and of the threat landscape, a set of algorithms is in charge of generating all the possible threat scenarios, simulating their outcomes evaluating the effect of countermeasures, and finally evaluating the likelihood and the potential impact of each outcome. In the end, the risk is calculated as a combination between likelihood and impact.
Limitations
Functionality [GAPM-RAS01] The proposed Risk Analysis methodology does not take into account cascading effect properly. This heavily limits the number of scenarios generated, as well as the level of detail of the impact simulation for each scenario. [GAPM-RAS02] The proposed Risk Analysis methodology does not allow to perform a cost-benefit analysis between costs related to protection (e.g. implementation of countermeasures) and reduction of risk.
Accuracy [GAPM-RAS03] If no wide dataset concerning threats is available, estimation of likelihood of threats can be inaccurate and, as a consequence, also outputs of Risk Analysis are not reliable. [GAPM-RAS04] Neglecting cascading effect could lead to a low level of accuracy of Risk Analysis outputs.
Scalability and computational [GAPM-RAS05] The methodology is scalable, but the computational time grows proportionally with the complexity of the system.
Deployment [GAPM-RAS06] The proposed Risk Analysis methodology needs a detailed modelling of the system under examination and a fine breakdown of the system into different components. [GAPM-RAS07] Wide datasets about past threats and incidents are needed to make the proposed Risk Analysis methodology work properly.

Learning curve No relevant gap or limitation has been identified.
Lack of automation [GAPM-RAS08] Modelling of the system under examination should be done manually by the user.
Reference environment [GAPM-RAS09] The proposed Risk Analysis methodology has TRL 6 and it can be implemented in prototypal application.
Costs [GAPM-RAS10] The proposed Risk Analysis methodology has a computational time proportional to the size of the system under examination, therefore in case of complex systems, costs of computation could be high.
Standards [GAPM-RAS11] Risk Analysis proposed methodology does not fulfil any standard.
References

3.6.1.7 Vulnerability Analysis of Cryptographic Modules Against Hardware-Based Attacks

Name: **Vulnerability analysis of cryptographic modules against hardware-based attacks**
Short description According to Kerckhoff's hypothesis [VAC1], it is assumed that the overall security of any cryptographic system depends entirely on the security of the key and all other parameters of the crypto system are public. According to this hypothesis, encryption algorithms are assumed to be open as long as the key generation scheme is not secure. Vulnerability analysis is complementary to cryptography. The strength of a cryptosystem depends on the key used, or in other words, the attacker's ability to predict the key. In this method, a 4-step key security evaluation method is proposed.
Limitations
Functionality [GAPM-VAC01] Only chaotic and ring oscillatory-based RNGs can be analysed for vulnerability.
Accuracy [GAPM-VAC02] The probability that the bits of ring oscillator RNGs can be predicted as a result of vulnerability analysis is 50 % and above. This ratio is 100 % for chaotic oscillator-based RNGs.
Scalability and computational

[GAPM-VAC03] Vulnerability analysis of ring oscillator-based RNGs can only be made at the FPGA design stage.
Deployment [GAPM-VAC04] For vulnerability analysis of ring oscillator-based RNGs, FPGA design IDEs and many numerical analysis tools can be used. Also, numerical analysis tools are required for analysis of chaotic oscillator-based RNGs (DynamicSolver, Matlab etc.).
Learning curve [GAPM-VAC05] Lack of standardised interoperability scheme in deployment of the proposed method) Since the provided method has many different interpretations in different contexts, the viewpoint taken in Valu3S is that an interoperable and generic model is needed to incorporate the interoperability of the presented V&V method with all systems at deployment phase. There is still a lack of a standardised interoperability scheme to deploy both hardware and software components of the proposed V&V method. This gap can be mitigated by context-specific analysis of integration and deployment requirement particular for the operational goals of the use case.
Lack of automation [GAPM-VAC06] The method requires effective and qualified human intervention. It is not applicable on systems operating in the field.
Reference environment [GAPM-VAC07] This method can only be applied in simulation environment for ring oscillator-based RNGs. For chaotic oscillator-based RNGs, analyses can be made on the prototype in lab environment.
Costs No relevant gap or limitation has been identified.
Standards No relevant gap or limitation has been identified.
References
• [VAC1] K. Martin, Everyday Cryptography: Fundamental Principles and Applications, 2nd Edition, Oxford University Press,2017.

3.6.1.8 Wireless interface network security assessment

Name: **Wireless interface network security assessment**
Short description Both radio link and Wi-Fi interfaces, which are normally used by the authorized teleoperator to send commands and other instructions to the Robotic Control Unit, can be also exploited for malicious purposes thanks to its direct connectivity with the CANBUS. The aim of this method is twofold:

1. Radio link interface: the radio link communications can be based on proprietary protocols, making its security level evaluation difficult. Software-Defined Radio (SDR)-based solutions can be used to this aim, monitoring the activity on the teleoperation radio link, capturing and replying commands, and even acting as fake controllers.
2. Wi-Fi interface: other CANBUS-related attacks can exploit the Wi-Fi interface. This method aims to focus on the possible harms due to this kind of attacks considering the rightful use of this interface. For example, the consequences of injection of packets containing modified maps directly on the CANBUS through the Wi-Fi interface.

Limitations

Functionality

[GAPM-WIN01] There could be possible limitations depending on the frequency bands used for the radio link communications and on the hardware and CANBUS network segmentation.

Accuracy

[GAPM-WIN02] Different factors could limit the accuracy, such as the power and the signal to noise ratio of the signal received by the radio link interface and how close to the attacker system the malicious user can stay and how long.

Scalability and computational

No particular restrictions for scalability. No need for particular computational resources.

Deployment

No limitations that could hinder the method's employment in a real-world context.

Learning curve

No relevant gap or limitation has been identified.

Lack of automation

[GAPM-WIN03] The method requires human intervention to iterate the method, at least at the first stage. Different attacks can be employed and each of them could need a "tuning" phase to better adapt and be effective to the attacked system.

Reference environment

[GAPM-WIN04] The method can be applied in simulation, emulation, and prototype environments. Its possible application in TRL6-7 environment could not be straightforward and require a further integration effort.

Costs:

No relevant gap or limitation has been identified since no huge investments are required.

Standards

No relevant gap or limitation has been identified.
References

3.6.2 General Semi-Formal Analysis

3.6.2.1 Code design and coding standard compliance checking

Name: **Code design and coding standard compliance checking**
Short description Coding standards and coding rules, agreed at design stages and employed both at design and development phases, facilitate the verifiability of produced code. Design of the code based on modularization: a method to organize large programs in smaller parts, i.e., the modules. Every module has a well-defined interface toward client modules that specifies how provided functionalities are made available. Moreover, every module has an implementation part that hides the code and any other private implementation detail the client modules should not care of with private data protection and interface.
Limitations
Functionality [GAPM-CDC01] Automation in this method is not applicable, as the method itself relies on the coding skills of the programmers. However, the application of standard coding rules and design patterns leads to a fast and proven development of the code by different programmers.
Accuracy [GAPM-CDC02] The accuracy of the method is related on the knowledge and continuous application of the rules by the software designers and developers.
Scalability and computational No relevant gap or limitation has been identified.
Deployment [GAPM-CDC03] Style-specific rules may not be covered by tools
Learning [GAPM-CDC04] Depending on the code design and code rules, the curve can be gentle (i.e. takes a slightly large amount of time) but it may not need high-level skills for coding; in fact to some extent, the lower the coding skill, the better to comply to the coding rules.
Lack of automation [GAPM-CDC05] Little automations are feasible.
Reference environment

No relevant gap or limitation has been identified.
Costs [GAPM-CDC04] Costs related are the training of the employees and the practice related. Other costs may be estimated by the time spent in the execution of walkthrough verification of the code.
Standards No relevant gap or limitation has been identified.
References

3.6.2.2 Knowledge-centric system artefact quality analysis

Name: **Knowledge-centric system artefact quality analysis**
Short description Method to assess the quality of systems artefacts (e.g. textual requirements specifications and system models) by exploiting knowledge bases, e.g. an ontology [KCQ1]. The assessment is quantitative according to different artefact characteristics (correctness, consistency, and completeness) and to different metrics (e.g. based on the number of elements with a given property in an artefact, such as the number of vague words in a requirement).
Limitations
Functionality [GAPM-KCQ01] The amount of model-specific quality analysis means is currently limited. Most of the available support focuses on textual requirements.
Accuracy [GAPM-KCQ02] A detailed study of quality analysis accuracy has not been conducted. Nonetheless, the practical experience and the feedback from the users suggest that the accuracy is suitable.
Scalability and computational [GAPM-KCQ03] Issues can arise with large and complex system artefacts. Tool solutions have nonetheless been developed to mitigate it.
Deployment [GAPM-KCQ04] Connectors with the system artefact sources are required.
Learning curve [GAPM-KCQ05] There is a barrier in the need for knowing how to create and manage ontologies.
Lack of automation [GAPM-KCQ06] Creation and management of ontologies is mostly a manual effort that can require significant time.

Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-KCQ07] Creation and management of ontologies is mostly a manual effort that can require significant time.
Standards [GAPM-KCQ08] No explicit and direct link with compliance has been established for most assurance/engineering standards. Nonetheless, the method (1) has been applied for many systems under regulatory requirements and (2) supports INCOSE rules for writing for writing requirements, among other reference documents.
References
• [KCQ1] Parra, E., Alonso, L., Mendieta, R., de la Vara, J.L.: Advances in Artefact Quality Analysis for Safety-Critical Systems. 30th International Symposium on Software Reliability Engineering (ISSRE 2019)

3.6.2.3 Knowledge-Centric Traceability Management

Name: **Knowledge-Centric Traceability Management**
Short description V&V method to manage the relationships between system artefacts, and thus how the system lifecycle has evolved and whether it has been adequate, that exploits knowledge representations in the form of ontologies for trace creation, management, and discovery, among other tasks.
Limitations
Functionality [GAPM-KCT01] Several activities of the traceability management process [KCT1] are not supported.
Accuracy [GAPM-KCT02] Accuracy has not been formally assessed.
Scalability and computational [GAPM-KCT03] Issues can arise with large and complex system artefacts. Tool solutions have nonetheless been developed to mitigate it.
Deployment [GAPM-KCT04] Connectors with the system artefact sources are required.
Learning curve [GAPM-KCT05] There is a barrier in the need for knowing how to create and manage ontologies.

Lack of automation No relevant gap or limitation has been identified.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-KCT06] Creation and management of ontologies is mostly a manual effort that can require significant time.
Standards [GAPM-KCT07] No explicit and direct link with compliance has been established for most assurance/engineering standards. Nonetheless, the method has been applied for systems under regulatory requirements.
References
• [KCT1] Cleland-Huang, J., Gotel, O. and Zisman, A. (eds.): Software and systems traceability. Heidelberg, Springer. 2012

3.6.2.4 Model-based assurance and certification

Name: **Model-based assurance and certification**
Short description Method that supports activities explicitly and directly targeted at system assurance and certification, e.g. management of compliance with standards, of assurance cases, and of assurance evidence. The method uses model-based technologies to facilitate the activities and ensure their suitability, such as correct and complete collection of assurance information.
Limitations
Functionality [GAPM-MAC01] The method could be further integrated with others for further assurance information analysis and collection.
Accuracy No relevant gap or limitation has been identified.
Scalability and computational [GAPM-MAC02] Depending on how the method is enacted, issues can arise with the management of large models.
Deployment

[GAPM-MAC03] The method most often requires tailoring (selection of activities needed) to specific companies and projects.
[GAPM-MAC04] Integration with further methods and tools can be needed in practice, e.g. with further means for model-based systems engineering.
[GAPM-MAC05] The support for workflow configuration is limited.
[GAPM-MAC06] Tool support usability can be improved.

Learning curve
No relevant gap or limitation has been identified.

Lack of automation
[GAPM-MAC07] The level of automation in assurance information management could be higher for certain tasks, e.g. for assurance information traceability management and verification.

Reference environment
[GAPM-MAC08] The application in real projects is limited for several elements of the method.
[GAPM-MAC09] The method has not been used in some domains, e.g. healthcare.

Costs
No relevant gap or limitation has been identified.

Standards
[GAPM-MAC10] The method explicitly supports compliance with standards. However, compliance management for some domains and standards has not been conducted yet.

References

- [MAC1] de la Vara, J.L., Ruiz, A., Gallina, B., Blondelle, G., Alaña, E., Herrero, J., Warg, F., Skoglund, M., Bramberger, R.: The AMASS Approach for Assurance and Certification of Critical Systems. embedded world Conference 2019

3.6.2.5 Model-based Design Verification

Name: **Model-based Design Verification**

Short description

Model-Based Design (MBD) provides a basis for machine-assisted verification of the system under development by the definition of models, supporting initial design decisions as well as enabling early discovery of errors. Models are typically described in high-level languages, either standardized or proprietary, and can be analysed with different MBD Verification methods. When the models are defined with formal semantics, formal methods can be applied for a rigorous automatic analysis of system properties.

Limitations

Functionality

[GAPM-MBD01] Existing methodologies and tools may need to be extended depending on the specific domain of the system under development.
Accuracy [GAPM-MBD02] A potential gap between the system-level analysis and the real system could arise due to the fact that formal analysis may require to abstract away some critical aspects about both the system under development and the external environment it will operate in.
Scalability and computational [GAPM-MBD03] The automated exhaustive analysis is subject to the state-explosion problem, which can prevent verification activities from scaling up to the complexity required in an industrial context.
Deployment [GAPM-MBD04] In some contexts, the methods may focus on high-level system views and are not integrated with the deployment to real-world contexts or are limited to few target platforms.
Learning curve [GAPM-MBD05] The method requires MBD skills, as well as expertise in the formalization of requirements. Formal methods have a steep learning curve, however the use of existing tools in a specific domain can make the formal specification process a much easier task.
Lack of automation [GAPM-MBD06] MBD Verification requires the formalization of natural language requirements, which is largely a manual step. However, the verification results are mostly automated.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-MBD07] Some MBD methods have only proprietary tool support with expensive licenses.
Standards No relevant gap or limitation has been identified.
References

3.6.2.6 Traceability Management for Safety Software

Name: **Traceability Management for Safety Software**
Short description Methodological method, that aims to formalize and make traceable the decisions taken during all the stages of the V-cycle process.

Limitations
Functionality No relevant gap or limitation has been identified.
Accuracy No relevant gap or limitation has been identified.
Scalability and computational No relevant gap or limitation has been identified.
Deployment [GAPM-TMS01] Proper tools may need docker or server instances available.
Learning curve [GAPM-TMS02] The curve is quite steep but needs constant application throughout the process.
Lack of automation [GAPM-TMS03] Some automations are feasible.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-TMS04] Proper tools may be expensive in terms of license costs and at a lower extent for the hardware required.
Standards No relevant gap or limitation has been identified.
References

3.7 System-Type-Focused V&V

This group of methods tackles general or several V&V needs specific to certain system types, typically covering different V&V areas, e.g., formal verification and testing, that can be combined for system V&V as a whole. Therefore, this group complements the previous ones by presenting wider V&V needs for specific situations, e.g. in the scope of some VALU3S use case. In this sense, the methods below can correspond to larger aspects of V&V processes that need to be reviewed in the project to set their current status, including their strengths and limitations.

3.7.1 CPU Verification

Name: **Central Processing Unit (CPU) verification**
Short description CPU verification ensures that a CPU delivers services as intended. Several verification approaches and activities must be applied to reach this goal. The focus lies on open-source and free CPU verification methods.
Limitations
Functionality [GAPM-CPU01] A limitation is that a CPU cannot be verified for every possible software – state space is too large. [GAPM-CPU02] Free open-source tools for industry-grade CPU verification are currently under development (e.g. Open HW group RISC-V verification: https://github.com/openhwgroup/core-v-verif) [GAPM-CPU03] Some verification standards such as UVM are limited to block-level verification and are not easily extendable to the software layer.
Accuracy [GAPM-CPU14] Precise power consumption (for Differential Power Analysis Attacks mitigation) hardly verifiable on CPU models (e.g. RTL simulation). [GAPM-CPU04] Signal-integrity analysis has limited accuracy (e.g. maximum frequency for stable operation might differ for each produced CPU). [GAPM-CPU05] Simulation-based methods do not provide a proof of correct behaviour if not exhaustively exercised. For safety-critical applications, a combination with formal methods should be applied.
Scalability and computational [GAPM-CPU06] Higher accuracy (cycle-accurate verification and below, e.g. gate-level simulation) demands high computation power and limits the testing capacity for software.
Deployment [GAPM-CPU07] Industry-grade open-source tools not available yet. [GAPM-CPU08] Open-source tools do not provide a production verification environment. At integration, verification test programs have to be defined to test the desired functionality.
Learning curve [GAPM-CPU09] CPU verification demands highly skilled verification engineers (testing, simulation, formal verification, etc.)
Lack of automation [GAPM-CPU10] Traceability of requirements to design and implementation artifacts. Requirements and artifacts should be linked with tool support.

[GAPM-CPU11] Tool-assisted generation of formal statements (assertions, properties) from a textual/graphical specification.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-CPU15] CPU verification is very expensive (lots of hardware resources needed since CPUs have a high state space – lots of testcases necessary, tools might be expensive, lots of human resources needed) [GAPM-CPU12] Tool licenses become a dominating cost factor. [GAPM-CPU13] Modern SoC emulators are expensive on purchase and operation.
Standards No relevant gap or limitation has been identified.
References

3.7.2 Penetration Testing

Name: **Penetration Testing**
Short description Analysis of data corruption in communication between server, PLC and sensors.
Limitations
Functionality [GAPM-DMD01] Cybersecurity approaches have a fundamental weakness which is that some security vulnerabilities cannot be detected like "Zero Day Attacks". But there is no alternative healthy solution for penetration testing. Penetration testing must be applied by someone, who is outside the company, in order to get information from a different point of view in security.
Accuracy [GAPM-DMD02] The method does not provide a perfect security level. But it is enough for not being the weakest link in the chain. Sometimes false positive and false negative results can be achieved, thus it is required to double check the test and system. Additionally, a full-force attack may have some damage risk so that the penetration tests may be executed in lighter fashion.
Scalability and computational [GAPM-DMD03] Penetration test services need to access some base system at the implementation area (i.e. physical and virtual access). For a large wide area network, some regional administrators may not allow access to their domains so there might be some scalability issues not originating from the method itself but because of operational decisions. From a computational point of view no gaps

or limitations have been identified since the resources to perform penetration testing is considerably neglectable and a powerful PC will be enough to carry out the test appropriately.
Deployment No relevant gap or limitation has been identified: deployment of the method is not complicated since only network access is needed to perform the test. Base of penetration testing tools and methods are usually well-known.
Learning curve [GAPM-DMD05] The testing approach used requires some solid background knowledge. Actually, since the cyber-attacks evolve by the time, it is important to be up to date by continuously following state-of-the-art attacks. Hence, proper implementation of this attack requires high-level skills.
Lack of automation [GAPM-DMD06] Performance of penetration testing always depends on performance of penetration tester. We standardize the system using appropriate methods. However, it is not possible to fully automate these tests.
Reference environment No relevant gap or limitation has been identified. The tests will be carried out in the operational environment (TRL-7).
Costs No relevant gap or limitation has been identified since open-source software could be used. The main cost is originating from human resources.
Standards No relevant gap or limitation has been identified. The method fulfils ISO/IEC 27002:2013, which gives guidelines about organizational information security standards and information security management practices including the selection, implementation and management of controls taking into consideration the organization's information security risk environment(s).
References

3.7.3 Failure Detection and Diagnosis (FDD) in Robotic Systems

Name: **Failure Detection and Diagnosis (FDD) in Robotic Systems**
Short description Failure Detection and Diagnosis is a data-driven process monitoring method for testing and minimising downtimes, increasing the safety of operations, and reducing manufacturing costs. FDD comprises different approaches that can be distinguished into data-based, model-based, and knowledge-based approaches.

Limitations
Functionality [GAPM-FDD01] Data-based resp. data-driven approaches rely on the gained data in order to be able to extract useful information for failure detection and diagnosis. Thus, quantity and quality of the data is of utmost importance. Model-based approaches are totally dependent on the knowledge about the system in advance. The related diagnostic process relies on an explicit model of the normal system behaviour, its structure, and/or its known faults. [GAPM-FDD02] Knowledge-based approaches typically mimic the behaviour of a human expert and can combine model-based and data-driven approaches in a hybrid FDD approach, i.e., the failure can be detected through data analytics which is associated to a diagnosis afterwards by the model-based approach. The challenge with knowledge-based approach is the detection of unknown faults.
Accuracy [GAPM-FDD03] Robotic systems are typically highly dynamic and, if not completely covered, act in an uncertain physical environment. They may interact with objects, other robots, and humans and consequently carry a high degree of uncertainty, where unexpected outcomes might lead to unknown faults and failed interactions. An FDD technique is supposed to detect failures and to distinguish between failed interactions that resulted from internal faults and failed interactions that resulted from exogenous events. Regarding Human-Robot Interaction (HRI) in UC4, FDD is mainly concerned with safety such as safety protocols and standards, risk assessment techniques and collision avoidance. The usage of FDD in this kind of scenarios is still new and cover some uncertainties and inaccuracies which must be tackled.
Scalability and computational [GAPM-FDD04] Regarding and depending on data-driven model testing, the data volume will affect the scalability and computational performance. In fact, the number of, e.g., sensors that are used for process monitoring and data gathering will affect the required computational resources.
Deployment [GAPM-FDD05] FDD in HRI context is a relatively new field of application, thus it will have to cope with early adoption challenges, adaptations, connectivity, accessibility, etc.
Learning curve [GAPM-FDD06] FDD requires high-level skills as it can comprise and apply a variety respectively a combination of approaches at the same time for monitoring and testing. In case of e.g. applying data-driven machine learning, expert knowledge is required to identify and select the most accurate technique based on the available data (data volume, velocity, variety, veracity).
Lack of automation [GAPM-FDD07] The method requires some intervention and effort, e.g., for modelling involved devices and/or humans to a required detail for simulating real world behaviour for testing.

Reference environment [GAPM-FDD08] FDD can be applied in various environments and is not limited to a single or small group of it. However, it has to be adapted to the particularities of the scenario.
Costs [GAPM-FDD09] As there will be always effort for e.g., adaptation, modelling, data collection, etc., carried out by an expert or group of experts, a mid to high investment need to be considered.
Standards [GAPM-FDD10] Some gaps have been pointed out in all the identified FDD-related standards as the applicability of the method in robotics is not yet broadly validated. ISO TS 15066 defines safety requirements for collaborative industrial robot systems and working environment. In order to fulfil these, FDD has to define and use specific test scenarios taking into account the specific safety requirements which is a relatively new field of application. The standard supplements the requirements and instructions for the operation of collaborative industrial robots and robot systems listed in ISO 10218-1 and ISO 10218-2. Same applies for the related methods mentioned in D3.1 and which FDD does not directly / namely consider: - EN ISO 13849 Safety of machines - safety-related parts of control systems - EN ISO 12100 Safety of machinery - General principles for design - Risk assessment and risk reduction - EN 61508-1 Functional safety of electrical/electronic/programmable electronic safety-related systems
References

3.7.4 Model-Based Formal Specification and Verification of Robotic Systems

Name: **Model-Based Formal Specification and Verification of Robotic Systems**
Short description The software of autonomous robot systems is generally complex and safety critical. Model-checking, which is a powerful technique for software verification, can verify the safety requirements of robotic systems in the early design stages. With the combination of runtime verification, the verification coverage can be improved by ensuring the safety of the robotic system during execution.
Limitations
Functionality [GAPM-MBF01] Creating a model is very difficult and time-consuming. It needs to be facilitated. [GAPM-MBF02] The method requires some components to verify robotic systems, such as checking compliance with safety standards and checking collisions.

Accuracy The method is accurate if the system was modelled correctly and the system software was developed according to the model.
Scalability and computational [GAPM-MBF03] When robotic system software becomes too complicated, the state space expands, and consequently, many computation resources and long waiting times for verification are required.
Deployment [GAPM-MBF04] There are case studies in the literature in which this method is used in robotic systems, but there is no evidence for its use in industry.
Learning curve [GAPM-MBF05] The robotic system software must be formally specified and modelled with a formal language to verify the system by model checking. It requires expertise and experience gained during a long learning period.
Lack of automation [GAPM-MBF06] All system models and specifications are created manually by human users.
Reference environment No relevant gap or limitation has been identified.
Costs [GAPM-MBF07] There are both free and paid model-checker software tools. A software tool cost can be incurred if a paid one is chosen. [GAPM-MBF08] If the state spaces of the models are too large, there may be a need for powerful computers to reduce the verification time, causing a hardware cost. [GAPM-MBF09] The creation of the models takes time and requires experience. In this case, human resources will also constitute a cost item.
Standards [GAPM-MBF10] The method does not have tools that consider robot safety standards ((ISO 10218-1, ISO 10218-2, ISO/TS 15066) holistically.

Chapter 4 Gaps Overview

In this chapter an overview of the gaps is presented, highlighting which are the most frequent category of gaps found in the analysed methods.

Overall, 400 gaps have been identified as summarized in Table 4.1. Notice that some methods (namely *Behaviour Driven Model Development and Test-Driven Model Review* and *Model-based Mutation Testing*) are inheriting gaps from Model-based Testing: these gaps are counted in this table and in the following reports only with regard to Model-based Testing.

Table 4.1: The number of identified gaps for each method

Method	Category	# of gaps
Model-Implemented Attack Injection	Attack Injection	12
Simulation-based Attack Injection at System-level	Attack Injection	11
Vulnerability and Attack Injection	Attack Injection	11
Fault Injection in FPGAs	Fault Injection	5
Interface Fault Injection	Fault Injection	6
Model-Based Fault Injection for Safety Analysis	Fault Injection	6
Model-Implemented Fault Injection	Fault Injection	11
Simulation-based Fault Injection at System-level	Fault Injection	13
Software-Implemented Fault Injection	Fault Injection	8
Simulation-Based Robot Verification	Simulation	6
Simulation-Based Testing for Human-Robot Collaboration	Simulation	4
Test Optimization for Simulation-Based Testing of Automated Systems	Simulation	3
Virtual Architecture Development and Simulated Evaluation of Software Concepts	Simulation	13
Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance	Simulation	8
V&V of Machine Learning-Based Systems Using Simulators	Simulation	13
Behaviour-Driven Model Development and Test-Driven Model Review	Testing	5

Method	Category	# of gaps
Assessment of Cybersecurity-Informed Safety	Testing	9
Machine Learning Model Validation	Testing	4
Model-Based Mutation Testing	Testing	2
Model-Based Robustness Testing	Testing	3
Model-Based Testing	Testing	15
Risk-Based Testing	Testing	4
Signal Analysis and Probing	Testing	4
Software Component Testing	Testing	5
Test Parallelization and Automation	Testing	7
Dynamic Analysis of Concurrent Programs	Runtime Verification	8
Runtime Verification Based on Formal Specification	Runtime Verification	11
Test Oracle Observation at Runtime	Runtime Verification	6
Deductive Verification	Formal Source Code Verification	7
Source Code Static Analysis	Formal Source Code Verification	11
Behaviour-Driven Formal Model Development	General Formal Verification	4
Formal Requirements Validation	General Formal Verification	7
Model Checking	General Formal Verification	5
Reachability-Analysis-Based Verification for Safety-Critical Hybrid Systems	General Formal Verification	9
Theorem Proving and SMT Solving	General Formal Verification	7
Human Interaction Safety Analysis	SCP-Focused Semi-Formal Analysis	7
Intrusion Detection for WSN based on WPM State Estimation	SCP-Focused Semi-Formal Analysis	3
Kalman Filter-Based Fault Detector	SCP-Focused Semi-Formal Analysis	7

Method	Category	# of gaps
Model-Based Safety Analysis	SCP-Focused Semi-Formal Analysis	9
Model-Based Threat Analysis	SCP-Focused Semi-Formal Analysis	7
Risk Analysis	SCP-Focused Semi-Formal Analysis	11
Vulnerability Analysis of Cryptographic Modules against Hardware-based Attacks	SCP-Focused Semi-Formal Analysis	7
Wireless Interface Network Security Assessment	SCP-Focused Semi-Formal Analysis	4
Code Design and Coding Standard Compliance Checking	General Semi-Formal Analysis	6
Knowledge-Centric System Artefact Quality Analysis	General Semi-Formal Analysis	8
Knowledge-Centric Traceability Management	General Semi-Formal Analysis	7
Model-Based Assurance and Certification	General Semi-Formal Analysis	10
Model-Based Design Verification	General Semi-Formal Analysis	7
Traceability Management for Safety Software	General Semi-Formal Analysis	4
CPU Verification	System-Type-Focused V&V	15
Penetration Testing	System-Type-Focused V&V	5
Failure Detection and Diagnosis (FDD) in Robotic Systems	System-Type-Focused V&V	10
Model-Based Formal Specification and Verification of Robotic Systems	System-Type-Focused V&V	10
Total number of gaps		**400**

On average, 7.5 gaps or limitations have been identified for each method. The methods with the highest number of gaps or limitations (15) are *Model-based Testing* and *Penetration Testing*. The category with the highest number of highlighted gaps or limitations is Attack Injection, whose methods have 11.3 gaps or

limitations on average. The list of categories with the associated average number of gaps is shown in Table 4.2.

Table 4.2: Average number of gaps for each category of methods

Category	# of gaps
Attack Injection	11.3
Fault Injection	8.1
Simulation	7.8
Testing	5.8
Runtime Verification	8.3
Formal Source Code Verification	9.0
General Formal Verification	6.4
SCP-Focused Semi-Formal Analysis	6.9
General Semi-Formal Analysis	7.0
System-Type-Focused V&V	10.0

The type of gap with the highest number of occurrences is Functionality, about which 69 gaps have been identified. The complete list of type of gaps and the relative number of occurrences are shown in Table 4.3.

Table 4.3: Total number of gaps for each type of gap.

Type of gap	# of gaps
Functionality	69
Accuracy	56
Scalability	49
Deployment	47
Learning Curve	45
Reference Environment	25
Costs	50
Lack of Automation	43
Standards	16

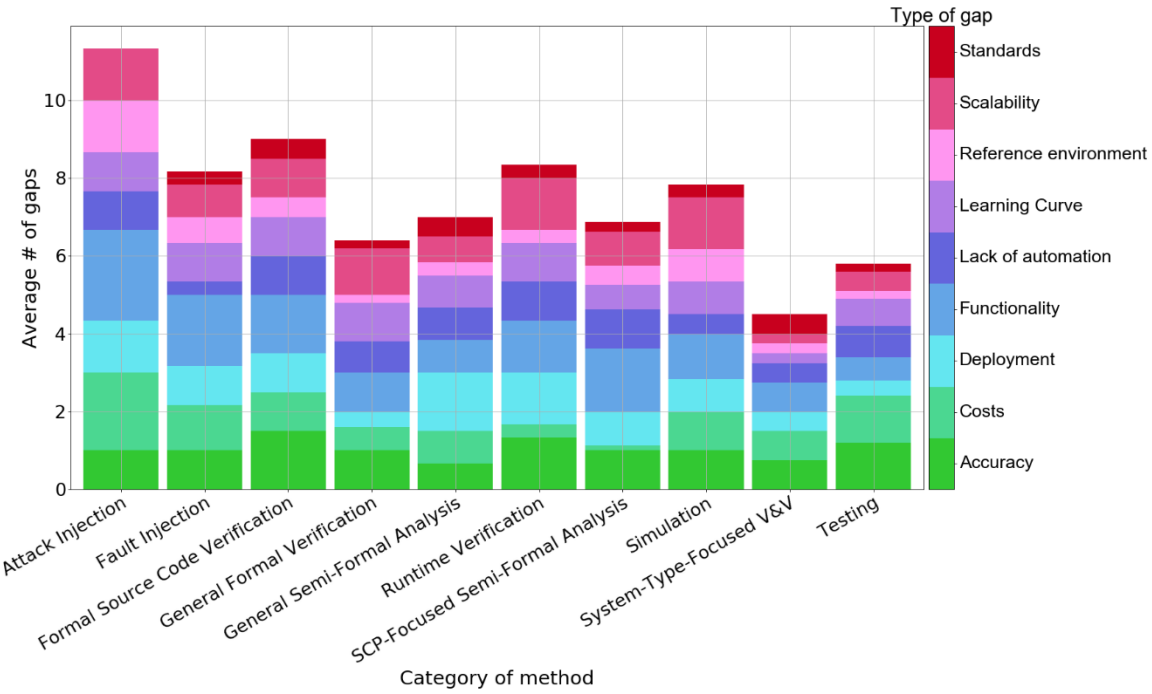


Figure 4.1: The average number of gaps per category of method and type of gap.

Figure 4.1 reports the average number of gaps for each category of method and for each type of gap. For example, Attack Injection methods include about 12 gaps on average overall: about 1 regards Accuracy, about 2 regard Costs etc… This allows to qualitatively understand which type of gap are more frequent in each category. For example, Testing methods have more frequently gaps or limitations about Accuracy, while fewer limitations were pointed out as regards Functionality.

Of course, these indications could be further analysed during the next activities of VALU3S (in particular, Task 3.3) because the detected gaps and limitations are going to drive the development of improved or new methods. So, understanding if some methods share the same type of limitations and which type of limitations are more frequent could clarify the direction where the improvements should be carried out.

Chapter 5 Tool-related Gaps and Limitations

Besides the gaps and limitations of the methods, it is worth highlighting that many applications cannot be faced with state-of-the-art V&V methods due to the lack or inadequacy of tools supporting such methods. As a matter of fact, a tool should implement all the functionalities of a method, should achieve high performances also on complex problems and should allow an easy and quick deployment in different real-world contexts. Moreover, since tools are supposed to be used by people with different technological skills, the ease of use and, in general, the user experience are important topics, too. For this reason, this section is focused on highlighting which methods tools are currently available for the methods analysed in Chapter 3 and the relative limitations. The focus here is not on the limitations of the methods but rather on the limitations in the implementation of the methods. Also, for tools, gaps and limitations have been labelled using the prefix "GAPT". Table 5.1 reports the results of this analysis. Notice that the table reports only the tools for which some gaps or limitations have been identified. Of course, other tools are available for each method but they are not reported here. A more complete report of available tools can be found in [1].

Table 5.1: List of the tools associated with different methods and the relative gaps and limitations.

V&V method	Tool	Limitations / gaps
Simulation-based Fault Injection at System-level.	Fault and Attack insertion in SUMO (Simulation of Urban MObility)	[GAPT01] The tool is limited to inject faults in SUMO. [GAPT02] Test configuration and results analysis are done manually. [GAPT03] No support for pre- and post-injection analysis.
Behaviour-Driven Formal Model Development	Gherkin	[GAPT04] Gherkin integration to ProB is only prototypical.
Behaviour-Driven Formal Model Development	MoMuT::Event-B	[GAPT05] MoMuT for Event-B is a prototype tool that does support only a subset of Event-B.
Deductive Verification	VCC	[GAPT07] No integration with other V&V/development tools, e.g. Simulink.
Deductive Verification	Frama-C	[GAPT08] No integration with other V&V/development tools, e.g. Simulink.

V&V method	Tool	Limitations / gaps
Deductive Verification	Verifast	[GAPT09] No integration with other V&V/development tools, e.g. Simulink.
Deductive Verification	Dafny	[GAPT10] No integration with other V&V/development tools e.g. Simulink.
Dynamic Analysis of Concurrent Programs	ANaConDA	[GAPT11] Missing support of some programming features (e.g., multi-process programs). [GAPT12] Some forms of properties to be checked not supported in sufficient generality (e.g., contracts of concurrency are not supported in a way distinguishing values of parameters of the functions involved in the contracts).
Failure Detection and Diagnosis (FDD) in Robotic Systems	CIROS Studio	[GAPT13] The model library does not include all types of devices needed in the simulation of some applications. [GAPT14] Human model needs enhancements.
Formal Requirements Validation	FRET	[GAPT15] It does not support first-order quantification.
General Formal Verification	Rodin	[GAPT16] Interactive proofs can be very time consuming.
Knowledge-Centric System Artefact Quality Analysis	RQA - Quality Studio	[GAPT17] The amount of available model-specific metrics is limited. [GAPT18] Some tools cannot be integrated.
Knowledge-Centric Traceability Management	Traceability Studio	[GAPT19] Some tools cannot be integrated.
Model Checking	Kind2	[GAPT20] Limited by state space explosion.
Model Checking	CoCoSim	[GAPT21] Limited integration with the other V&V/development tools, e.g. Simulink. Also, limited support for verifying detailed properties about arrays.

V&V method	Tool	Limitations / gaps
Model-Based Assurance and Certification	OpenCert	[GAPT22] There can be performance issues depending on how storage is configured. [GAPT23] Usability can be improved. [GAPT24] Support for workflow configuration is limited. [GAPT25] Further integration with external tools can be needed.
Model-based fault injection for safety analysis	xSAP	[GAPT26] Tool support usability can be improved, e.g., editing and customization of fault libraries.
Model-based fault injection for safety analysis	COMPASS	[GAPT27] Tool support usability can be improved, e.g., editing and customization of fault libraries.
Model-Based Formal Specification and Verification of Robotic Systems	ROS	[GAPT28] Needs integration with formal verification models
Model-Based Formal Specification and Verification of Robotic Systems	UPPAAL	[GAPT29] Creating models is difficult and requires expertise.
Model-Based Formal Specification and Verification of Robotic Systems	ROSRV	[GAPT30] It does not use formal models. It does not verify formally. [GAPT31] There are security problems.
Model-Based Formal Specification and Verification of Robotic Systems	GAZEBO	[GAPT32] It needs integration with formal verification methods
Model-Based Mutation Testing	Conformiq Designer	[GAPT33] Very limited set of mutations is provided.
Model-Based Robustness Testing	MoMuT	[GAPT34] The “fuzzing" features implemented are up to now rudimentary compared to fuzzing tools for e.g. protocol testing.
Model-Based Safety Analysis	CHESS FLA	[GAPT36] In presence of a large and complex systems, false FMEA rows may be generated and FT generation has to be improved.

V&V method	Tool	Limitations / gaps
Model-Based Testing	MoMuT	[GAPT37] Proprietary test case format.
Model-Implemented Attack Injection.	MODIFI (Matlab/Simulink based)	[GAPT38] Limited number and accuracy of attack models. [GAPT39] Limited support for pre- and post-injection analysis.
Model-Implemented Fault Injection.	MODIFI (Matlab/Simulink based)	[GAPT40] Limited number and accuracy of fault models. [GAPT41] Limited support for pre- and post-injection analysis.
Runtime Verification Based on Formal Specification	Copilot	[GAPT42] No integration with other V&V/development tools, e.g. Simulink.
Runtime Verification Based on Formal Specification	Spectra	[GAPT43] Missing support of automatic instrumentation of check points.
Signal Analysis and Probing	TestStand	[GAPT44] Proprietary data storage approach. Flexibility by e.g. SQL databases is needed
Simulation-based Fault and Attack Injection at System-level.	Fault and Attack insertion in SUMO (Simulation of Urban MObility)	[GAPT45] The tool is limited to inject attacks in SUMO. [GAPT46] Test configuration and results analysis are done manually. [GAPT47] No support for pre- and post-injection analysis.
Source Code Static Analysis	Frama-C	[GAPT48] Weak support of concurrency-related program issues.
Source Code Static Analysis	Facebook Infer	[GAPT49] Weak support of concurrency-related program issues, especially for low-level synchronisation.
Source Code Static Analysis	2LS	[GAPT50] Missing support for programs using certain kinds of data structures, such as arrays and hash tables.

V&V method	Tool	Limitations / gaps
Theorem Proving and SMT Solving	Z3	[GAPT51] Can time-out on complex properties.
V&V of Machine Learning-Based Systems Using Simulators	SUMO (Simulation of Urban Mobility)	[GAPT52] Only macro level simulations, not directly support for V&V of ML-based perception systems
V&V of Machine Learning-Based Systems Using Simulators	TASS/Siemens PreScan	[GAPT53] Limited support for diversity of scenarios.
V&V of Machine Learning-Based Systems Using Simulators	ESI Pro-SiVIC	[GAPT54] No comprehensive digital twin library.
V&V of Machine Learning-Based Systems Using Simulators	CARLA	[GAPT55] No comprehensive sensor models. [GAPT56] No comprehensive digital twin library
V&V of Machine Learning-Based Systems Using Simulators	Unreal Engine	[GAPT57] No comprehensive library of digital twin [GAPT58] Limitation in sensor models
V&V of Machine Learning-Based Systems Using Simulators	Environment Simulator Minimalistic (ESMINI)	[GAPT59] Not user friendly, written in C/C++. [GAPT60] Difficult to integrate it with machine learning software stack.
Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance	Unity 3D	[GAPT61] It is a closed-source software. Apart from that, the cost of deep customization of the user interaction can be high (development from scratch)
Vulnerability analysis of cryptographic modules against hardware-based attacks	Anadigm Designer2	[GAPT62] It is used to evaluate RNG designs, but requires high knowledge and experience. [GAPT63] Only analogue-based RNGs can be designed and evaluated. [GAPT64] Cannot work at high frequencies (max 1MHZ)

V&V method	Tool	Limitations / gaps
Vulnerability analysis of cryptographic modules against hardware-based attacks	Xilinx Vivado Design Suite	[GAPT65] It is used to evaluate RNG designs, but requires high knowledge and experience. [GAPT66] Only digital-based RNGs can be designed and evaluated.
Vulnerability analysis of cryptographic modules against hardware-based attacks	BigCrush	[GAPT67] Big Crush tests require at least 1TB of data. Thus, this test can only be implemented over strong computers
Vulnerability analysis of cryptographic modules against hardware-based attacks	DieHard	[GAPT68] Tests can be made by collecting the RNG output of the system to be evaluated as at least 1 million bits. This process takes time and requires extra hardware if it is not already integrated.
Vulnerability analysis of cryptographic modules against hardware-based attacks	NIST 800-22	[GAPT69] Tests can be made by collecting the RNG output of the system to be evaluated as at least 1 million bits. This process takes time and requires extra hardware if it is not already integrated.

Chapter 6 Use Case-related Gaps and Limitations

In this chapter, use cases are analysed to find the gaps in current V&V methods that prevent their application in real world scenarios. To reach this goal, the first step has been the identification of the state-of-the-art methods for the different use cases to understand what is currently used and which are the needed intervention to allow a better implementation of the scenarios. In particular, for each use case, the available methods have been divided into four groups:

1. Methods that are currently used in the use case.
2. Methods that will likely be used during the use case without any major development or improvement.
3. Methods that could be used in the use case after removing some limitations.
4. Methods that are not relevant for the use case, so they will not be used during the project for that use case.

This categorization allows one to identify where the effort should be put in place in the improvement and development of new methods. In particular, groups 1, 2 and 4 do not need any intervention, while methods in group 3 deserve some further analyses to understand which are the current limitations and how technology providers in WP3 should work to address such limitations.

It is worth noting that the definition of use cases is an ongoing process, so the definition of usable methods has been quite conservative: it is likely that not all the methods tagged as potentially usable will be actually used during the use case implementation. At the same time, the analysis presented in this chapter of the deliverable is necessarily partial, since not all the details of the demonstrators are known during this phase. Subsequent deliverables (in particular, all future deliverables of WP5 and D3.5 about new V&V methods) will address this topic more precisely. What is presented here is aimed only at providing some elements that could guide the work to develop new methods (Task 3.3, in particular).

Section 6.1—Section 6.12 report the results of these analysis for the 12 use cases. For each method, already identified issues are listed: for each of them it is specified between parentheses if it is a gap (i.e. something missing to allow the implementation of the method) or a limitation (i.e. some intrinsic limit of the method for that specific application). Notice that not all the use cases have some gaps or limitations reported. In these cases, the implementation of the use case is in an early stage and it is not possible to define use case driven limitations: of course, the gaps and limitations described for methods in Chapter 3 still hold true.

6.1 UC1 – Intelligent Traffic Surveillance

Currently Used Methods: Software Component Testing, Test Parallelization and Automation.

Methods that are planned to be used: Source Code Static Analysis.

Potentially usable methods: Simulation-based Attack Injection at System-level, Software-Implemented Fault Injection, V&V of Machine Learning-Based Systems Using Simulators, Model-Based Testing, Dynamic Analysis of Concurrent Programs, Penetration testing of Industrial Systems.

6.1.1 Identified Gaps and Limitations for Use Case Application

V&V of Machine Learning-Based Systems Using Simulators

- Availability of UC- related digital twin in simulator (scenes, scenarios, sensor models). Recreate the real-world scenarios of UC1 system in the simulator. (gap)
- V&V process that can use both real-world and simulation data. (gap)
- V&V of ML needs novel approaches as conventional techniques such as code reviews and coverage testing are only partly applicable. (gap)

Simulation-based Attack Injection at System-level

- A detailed description of the system from UC owner should be provided to apply this method. (gap)
- The availability of UC related digital twin in simulator (scenes, scenarios, sensor models) is needed for the method to work properly. (gap)

Penetration testing

- A detailed description of the system from UC owner should be provided to apply this method. (gap)
- The hardware/software related to the use case should be provided. (gap)
- Several variants of the method should be tried to improve its effectiveness, but, due to limited tools, time, scope, and access, not all of them will we explored. (limitation)

Dynamic Analysis of Concurrent Programs

- It requires one or more automatic tests (no assertion needed though) including execution environment or its test double (or the test harness). (gap)
- In the context of the use case, the method may provide a lot of false positives. (limitation)
- The method introduces significant overhead, so it cannot be used on SUT with real-time characteristic. (limitation)

6.2 UC2 – Car Teleoperation

Methods that are planned to be used: Software-Implemented Fault Injection, Assessment of Cybersecurity-Informed Safety, Dynamic Analysis of Concurrent Programs, Source Code Static Analysis, Penetration testing of Industrial Systems.

Potentially usable methods: Simulation-based Attack Injection at System-level, Simulation-based Fault Injection at System-level, Virtual Architecture Development and Simulated Evaluation of Software Concepts, Model-Based Testing, Human Interaction Safety Analysis

6.2.1 Identified Gaps and Limitations for Use Case Application

Simulation-based Attack Injection at System-level

- A proper tool to be applied in the context of the use case is currently to be identified. (gap)

Simulation-based Fault Injection at System-level

- The configuration of the Clumsy tool, currently used in the scope of the use case, is unsatisfiable. The development of a network monitoring tool to cover areas out of scope of Clumsy is needed. (gap)

Software-Implemented Fault Injection

- The configuration of the Clumsy tool, currently used in the scope of the use case, is unsatisfiable. (gap)
- The development of a network monitoring tool to cover areas out of scope of Clumsy is needed. (gap)

Source Code Static Analysis

- The number of checks is not sufficient for the target of the use case. Extension of Facebook Infer and/or Frama-C frameworks will be needed to support more common weakness or specific-purpose analyses (gap).

6.3 UC3 – Radar Systems for ADAS

Currently used methods: Fault Injection in FPGAs, Interface Fault Injection, Software-Implemented Fault Injection, Software Component Testing, Runtime Verification Based on Formal Specification, Source Code Static Analysis, Behaviour-Driven Formal Model Development, Formal Requirements Validation, Model Checking, Model-Based Safety Analysis, Model-Based Threat Analysis, Risk Analysis, Wireless Interface Network Security Assessment, Code Design and Coding Standard Compliance Checking, Model-Based Design Verification, Traceability Management for Safety Software

Methods that are planned to be used: Test Parallelization and Automation, Signal Analysis and Probing

Potentially usable methods: Model-Based Fault Injection for Safety Analysis, Model-Implemented Fault Injection, Risk-Based Testing

6.4 UC4 – Human-Robot-Interaction in Semi-Automatic Assembly Processes

Currently used methods: Model-Based Fault Injection for Safety Analysis, Simulation-Based Testing for Human-Robot Collaboration, Virtual Architecture Development and Simulated Evaluation of Software Concepts, Machine Learning Model Validation, Model-Based Testing, Model-Based Safety Analysis, Model-Based Threat Analysis, Failure Detection and Diagnosis (FDD) in Robotic Systems.

Methods that are planned to be used: Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance.

Potentially usable methods: Model-Implemented Attack Injection, Model-Implemented Fault Injection, Simulation-based Fault Injection at System-level, Software-Implemented Fault Injection, V&V of Machine Learning-Based Systems Using Simulators, Behaviour-Driven Model Development and Test-Driven Model Review, Assessment of Cybersecurity-Informed Safety, Formal Requirements Validation, Reachability-Analysis-Based Verification for Safety-Critical Hybrid Systems, Human Interaction Safety Analysis, Kalman Filter-Based Fault Detector, Model-Based Formal Specification and Verification of Robotic Systems.

6.4.1 Identified Gaps and Limitations for Use Case Application

Behaviour-Driven Model Development and Test-Driven Model Review

- The use case has not adopted a tool to sketch, generate and evaluate the UML diagrams. (gap)
- The methodology and the tool proposed seem to be more intended for the software development field. (limitation)
- The availability and encourage to use the referred tool have not been done properly. (limitation)
- The method seems to strongly rely into the proposed tool framework. (gap)

Formal Requirements Validation

- Formal specifications and requirements have not been defined for the use case. (gap)
- Currently no formal checkers or tools have been implemented in the use case. (gap)

Model-Based Formal Specification and Verification of Robotic Systems

- Property specifications has to be provided. (limitation)
- System level model has to be provided. (gap)
- The model checker is not clearly defined. (limitation)
- Adoption of an extra tool to apply the method. (limitation)

Model-Implemented Attack Injection

- Clearly define the fault models. (gap)
- Difficult to extend the fault injection methodology to the actual physical system. (limitation)

Model-Implemented Fault Injection

- Evaluate possible scenarios for application and define the attack models. (gap)
- Difficult to extend the attack injection methodology to the actual physical system. (limitation)

Simulation-based Fault Injection at System-level

- The system level model platform is currently under development. (gap)
- No knowledge about the proposed traffic simulators. (limitation)

6.5 UC5 – Aircraft Engine Controller

Currently used methods: Reachability-Analysis-Based Verification for Safety-Critical Hybrid Systems.

Methods that are planned to be used: Model-Implemented Attack Injection, Model-Implemented Fault Injection, Behaviour-Driven Formal Model Development, Formal Requirements Validation, Model Checking, Theorem Proving and SMT Solving, Model-Based Design Verification.

Potentially usable methods: Interface Fault Injection, Runtime Verification Based on Formal Specification, Deductive Verification, Source Code Static Analysis.

6.6 UC6 – Agricultural Robot

Methods that are planned to be used: Interface Fault Injection, Human Interaction Safety Analysis, Model-Based Safety Analysis, Wireless Interface Network Security Assessment.

Potentially usable methods: Simulation-Based Robot Verification, Assessment of Cybersecurity-Informed Safety, Software Component Testing, Code Design and Coding Standard Compliance Checking, Traceability Management for Safety Software.

6.6.1 Identified Gaps and Limitations for Use Case Application

Model-Based Safety Analysis - Failure Logic Analysis (FLA)

- Since the method has not been used in this context yet, the workflow and the different components that should be used is still to be defined. (gap)

Risk Analysis

- A dataset about threats and faults occurred to agricultural automated systems is currently not available, so gathering this information with the higher level of detail is required for applying the method. (gap)
- The modelling of each system component could be difficult to be performed with high granularity. Moreover, also the interdependencies among them, could be very complex and not fully described by a model. (limitation)
- Since it is a new application field, compliance to standard related to utilization of robots and automated systems in the agriculture domain should be addressed. (gap)

6.7 UC7 – Human-Robot Collaboration in a Disassembly Process with Workers with Disabilities

Methods that are planned to be used: Simulation-Based Testing for Human-Robot Collaboration, Test Optimization for Simulation-Based Testing of Automated Systems, Virtual & Augmented Reality-Based User Interaction V&V and Technology Acceptance.

6.8 UC8 – Neuromuscular Transmission for Muscle Relaxation Measurements

Methods that are planned to be used: Model-Based Safety Analysis, Risk Analysis.

Potentially usable methods: Software-Implemented Fault Injection.

6.8.1 Identified Gaps and Limitations for Use Case Application

Software-Implemented Fault Injection

- The modelling of stochastic processes in infusion pumps and TOF measurements is currently not available and needs to be developed to apply the model. (gap)
- A model for pseudo-random noise to infusion pump outputs and to TOF/PTC measurements need to be developed to apply the method. (gap)
- Some adaptation is needed to apply the method in the Infusion controller for NMT regulation. (gap)

6.9 UC9 – Autonomous Train Operation

Potentially usable methods: V&V of Machine Learning-Based Systems Using Simulators, Machine Learning Model Validation.

6.9.1 Identified Gaps and Limitations for Use Case Application

V&V of Machine Learning-Based Systems Using Simulators

- A complete dataset containing a variety of scenarios covering (almost) all possible working conditions should be developed to apply the method. While most simulators to generate this type of datasets is focused on automotive, no railway scenarios are currently achievable by means of them. (gap)
- The method is statistical, so cannot ensure sufficient reliability for critical applications and cannot fulfil standards in the railway domain, since they are based on deterministic applications. (limitation)

Machine Learning Model Validation

- Since fully auto labelled data in railway domain simulators is not available, a custom semi-automatic validation framework should be created, which requires huge investments in terms of budget and human resources. (gap)
- The method is statistical, so cannot ensure sufficient reliability for critical applications and cannot fulfil standards in the railway domain, since they are based on deterministic applications. (limitation)

6.10 UC10 – Safe Function Out-Of-Context

Methods that are planned to be used: Fault Injection in FPGAs, Model-Based Testing, Model Checking

Potentially usable methods: Interface Fault Injection, Model-Based Fault Injection for Safety Analysis, Software-Implemented Fault Injection, Software Component Testing, Runtime Verification Based on Formal Specification, Source Code Static Analysis, Formal Requirements Validation, Theorem Proving and SMT Solving, Model-Based Safety Analysis, Knowledge-Centric Traceability Management, Model-Based Assurance and Certification, Model-Based Design Verification, Traceability Management for Safety Software.

6.11 UC11 – Automated Robot Inspection Cell for Quality Control of Automotive Body-In-White

Currently Used Methods: Simulation-Based Robot Verification

Methods that are planned to be used: Model-Based Testing, Runtime Verification Based on Formal Specification, Model-Based Threat Analysis, Vulnerability analysis of cryptographic modules against hardware-based attacks, Penetration testing of Industrial Systems, Model-Based Formal Specification and Verification of Robotic Systems.

Potentially usable methods: Model-Implemented Attack Injection, Simulation-based Attack Injection at System-level, Vulnerability and Attack Injection, Interface Fault Injection, Model-Based Fault Injection for Safety Analysis, Software-Implemented Fault Injection, Software Component Testing, Human Interaction Safety Analysis

6.11.1 Identified Gaps and Limitations for Use Case Application

Virtual Architecture Development and Simulated Evaluation of Software Concepts

- Since the Simulation environments cannot simulate the accelerated movements, start and slowdown acceleration movements of the UC11 could not be able to be simulated. (gap)
- Body-in white system of UC11 works with PLCs in real world and simulation world does not have any perfection to simulate the PLC works. (gap)

6.12 UC13 – Industrial Drives for Motion Control

Methods that are planned to be used: Behaviour-Driven Model Development and Test-Driven Model Review, Model-Based Mutation Testing, Model-Based Robustness Testing, Model-Based Testing, Test Oracle Observation at Runtime, Model Checking, CPU Verification.

Potentially usable methods: Model-Based Fault Injection for Safety Analysis, Virtual Architecture Development and Simulated Evaluation of Software Concepts.

6.12.1 Identified Gaps and Limitations for Use Case Application

Virtual Architecture Development and Simulated Evaluation of Software Concepts

- Fault injection for changing the behavior of simulation components is not directly supported by current tools associated with the method, which would be needed for fault-injection analysis of motor position sensor data. (gap)
- The technical connection of FERAL to the simulation components of the UC (e.g., proprietary multi-physics simulator AMESim for motor modelling and simulation, QEMU, SystemC and (real-time) operating systems) needs to be implemented by means of proper connectors and adaptors. (gap)

Chapter 7 Conclusions

In this deliverable, gaps and limitations of V&V methods presented in D3.1 have been reviewed and analysed. Overall, 53 methods belonging to 13 groups (as defined in D3.1 [1]) have been analysed considering 9 categories of limitations. Considering all the categories, 400 gaps have been identified. The number of gaps per method ranges from 1 to 15, with an average of 7.5. The category with the highest average number of gaps or limitations is Attack Injection, whose methods have 11.3 gaps on average. The most frequent type of gap or limitation is Functionality: 69 of them have been pointed out. In addition, some gaps and limitations of tools have been considered, too, to highlight, besides the theoretical limitations, the practical issues that prevent actual deployment of V&V methods.

At last, also inputs from use cases have been considered, identifying what is missing in the state-of-the-art of V&V methods to allow an effective implementation of the VALU3S demonstrators. Since the implementation of use cases is an ongoing process, this picture is necessarily partial, but still, it is a good starting point to identify the direction of further developments in V&V methods. The limitations of methods together with the limitations derived from use cases will be the input for Task 3.3 since they will guide the improvements to existing methods and the development of new methods.

The results of this deliverable are in connection with the VALU3S objectives and KPIs. In particular, the identification of gaps and limitations is a prerequisite for Objective 2 (To overcome the SCP gaps and limitations of cyber-physical systems), Objective 5 (To suggest and validate new as well as state-of-the-art evaluation scenarios for safety, cybersecurity and privacy evaluation) and Objective 6 (To develop and improve V&V tools and evaluation criteria). Related to these objectives, this deliverable is contributing to achieve some of the project's KPIs, namely:

- Improve at least 14 V&V methods in order to create VALU3S repository of improved V&V methods (related to Objective 2).
- Present and detail at least 13 novel evaluation scenarios (including their requirement specifications) for safety, security and privacy evaluation through 13 realistic use cases (related to Objective 5).
- Improve and/or develop at least 24 V&V tools that aim to improve the time and cost of V&V processes while dealing with hardware-, software- and system-level cyber-physical risks (related to Objective 6).
- Incorporate and make use of at least 13 SCP evaluation criteria as well as at least 11 evaluation criteria suitable for measuring the level of improvement obtained in the V&V processes (related to Objective 6).

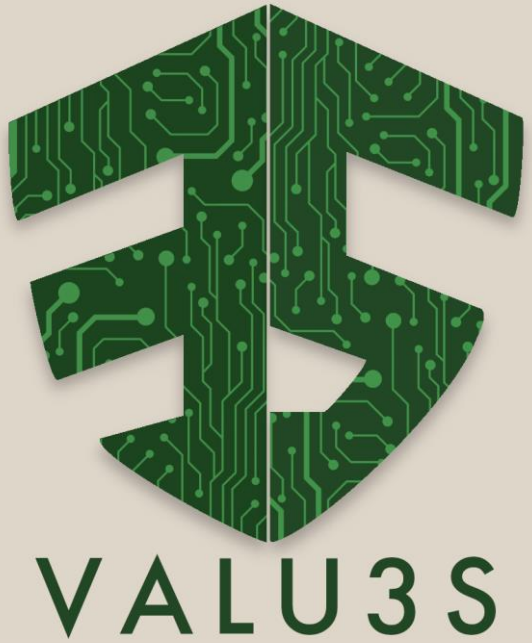
References

[1] VALU3S Consortium, “Deliverable D3.1 - V&V methods for SCP evaluation of automated systems,” 2020.

[2] VALU3S Consortium, “Deliverable D1.1 - Description of use cases as well as scenarios,” 2020.

[3] VALU3S Consortium, “Deliverable D1.2 - SCP requirements as well as identified test cases,” 2020.

[4] VALU3S Consortium, “Deliverable D5.1 - Initial Demonstration Plan and a List of Evaluation Criteria,” 2020.



www.valu3s.eu

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey.