A Proposal for the Classification of Methods for Verification and Validation of Safety, Cybersecurity, and Privacy of Automated Systems

Giann Nandi

Jose Luis de la Vara

Thomas Bauer

Bernhard Fischer

Mustafa Karaca

Henrique Madeira

Martin Matschnig

Silvia Mazzini

Fabio Patrone

David Pereira*

José Proença*

Rupert Sclick

Stefano Tonetta

Ugur Yayan

Behrooz Sangchoolie

*CISTER Research Centre

CISTER-TR-210905

2021/09/08

A Proposal for the Classification of Methods for Verification and Validation of Safety, Cybersecurity, and Privacy of Automated Systems

Giann Nandi*, Jose Luis de la Vara, Thomas Bauer, Bernhard Fischer, Mustafa Karaca, Henrique Madeira, Martin Matschnig, Silvia Mazzini, Fabio Patrone, David Pereira*, José Proença*, Rupert Sclick, Stefano Tonetta, Ugur Yayan, Behrooz Sangchoolie

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: giann@isep.ipp.pt, joseluis.delavara@uclm.es, thomas.bauer@iese.fraunhofer.de, bernhard.bf.fischer@siemens.com, mustafa.karaca@inovasyonmuhendislik.com, henrique@dei.uc.pt, martin.matschnig@siemens.com, silvia.mazzini@intecs.it, f.patrone@edu.unige.it, drp@isep.ipp.pt, pro@isep.ipp.pt, rupert.schlick@ait.ac.at, tonettas@fbk.eu, ugur.yayan@inovasyonmuhendislik.com, behrooz.sangchoolie@ri.se

https://www.cister-labs.pt

Abstract

As our dependence on automated systems grows, so does the need for guaranteeing their safety, cybersecurity, and privacy (SCP). Dedicated methods for verification and validation (V&V) must be used to this end and it is necessary that the methods and their characteristics can be clearly differentiated. This can be achieved via method classifications. However, we have experienced that existing classifications are not suitable to categorise V&V methods for SCP of automated systems. They do not pay enough attention to the distinguishing characteristics of this system type and of these quality concerns. As a solution, we present a new classification developed in the scope of a large-scale industry-academia project. The classification considers both the method type, e.g., testing, and the concern addressed, e.g., safety. Over 70 people have successfully used the classification on 53 methods. We argue that the classification is a more suitable means to categorise V&V methods for SCP of automated systems and that it can help other researchers and practitioners.

A Proposal for the Classification of Methods for Verification and Validation of Safety, Cybersecurity, and Privacy of Automated Systems

Jose Luis de la Vara[1], Thomas Bauer[2], Bernhard Fischer[3], Mustafa Karaca[4], Henrique Madeira[5], Martin Matschnig[3], Silvia Mazzini[6], Giann Spilere Nandi[7], Fabio Patrone[8], David Pereira[7], José Proença[7], Rupert Sclick[9], Stefano Tonetta[10], Ugur Yayan[4], and Behrooz Sangchoolie[11]

[1]Universidad de Castilla-La Mancha, Spain, [2]Fraunhofer, Germany, [3]Siemens, Austria, [4]Inovasyon Muhendislik, Turkey, [5]Universidade de Coimbra, Portugal, [6]Intecs, Italy, [7]ISEP, Portugal, [8]University of Genoa, Italy, [9]AIT, Austria, [10]FBK, Italy, [11]RISE, Sweden
joseluis.delavara@uclm.es, thomas.bauer@iese.fraunhofer.de, {bernhard.bf.fischer, martin.matschnig}@siemens.com, {mustafa.karaca, ugur.yayan}@inovasyonmuhendislik.com, henrique@dei.uc.pt, silvia.mazzini@intecs.it, {giann, drp, pro}@isep.ipp.pt, f.patrone@edu.unige.it, rupert.schlick@ait.ac.at, tonettas@fbk.eu, behrooz.sangchoolie@ri.se

Abstract. As our dependence on automated systems grows, so does the need for guaranteeing their safety, cybersecurity, and privacy (SCP). Dedicated methods for verification and validation (V&V) must be used to this end and it is necessary that the methods and their characteristics can be clearly differentiated. This can be achieved via method classifications. However, we have experienced that existing classifications are not suitable to categorise V&V methods for SCP of automated systems. The classifications do not pay enough attention to the distinguishing characteristics of this system type and of these quality concerns. As a solution, we present a new classification developed in the scope of a large-scale industry-academia project. The classification considers both the method type, e.g. testing, and the concern addressed, e.g. safety. Over 70 people have successfully used the classification on 53 methods. We argue that the classification is a more suitable means to categorise V&V methods for SCP of automated systems and that it can help other researchers and practitioners.

Keywords: Verification and Validation, V&V, method, classification, safety, cybersecurity, privacy, automated system.

1 Introduction

Automated systems such as industrial robots and advanced driving systems play an increasingly important role in society. They support many daily-life activities and we strongly depend on them. On the other hand, as the use and complexity of these systems are growing, system manufacturers and component suppliers require methods that help them to confirm that safety, cybersecurity, and privacy (SCP) requirements are satisfied

[Ref]. This is necessary so that the systems can be deemed dependable. From a general perspective, a method corresponds to a particular procedure for accomplishing or approaching something, especially a systematic or established one [Ref]; in the scope of this paper, for verification and validation (V&V) of automated systems. Examples of these methods are fault injection [Ref] and model-based testing [Ref].

The features of the new generation of automated systems require that dedicated V&V methods (usually a combination of methods) are applied to them [Ref]. The methods must consider how to cope with the scale and complexity of the systems, the aspects that make them cyber-physical, and their specific quality needs, among other issues. For example, the use of software-focused V&V methods alone is often not sufficient. This also implies that manufacturers and suppliers need to clearly distinguish among different V&V methods and their characteristics to be able to select the most adequate ones during a system's lifecycle. Method classifications can aid in this task.

However, when involved in the analysis and characterisation of V&V methods for SCP of automated systems, we have experienced that existing classifications are not suitable. Among the issues identified, the classifications do not pay enough attention to specific aspects such as the need for analysing possible faults and attacks at early development stages or for ascertaining what SCP aspect a given V&V method deals with. The descriptions of existing classification are also usually not clear enough to help users to decide upon how to best classify a V&V method and to select the most suitable method for a given V&V need. If these problems arise, then the selection and use of V&V methods for SCP of automated systems can be less effective, ultimately impacting the cost and dependability of a system.

We aim to address these issues by proposing a new classification of V&V methods. We have created it in the scope of VALU3S [Ref], a large-scale industry-academia project in which 41 partners from ten countries are cooperating towards improving how automated systems are verified and validated with respect to SCP requirements. Among the activities of the project, we identify, analyse, and classify methods that could improve V&V of specific industrial use cases from the automotive, agriculture, railway, healthcare, aerospace, and industrial automation domains.

The classification distinguishes between two main facets of a V&V method: the general method type and the concern addressed. For example, penetration testing [Ref] is a testing method for cybersecurity. Thanks to the classification, we have managed to classify tens of V&V methods and differentiate among them more precisely. Our initial aim in VALU3S was to reuse some existing classification, but we found issues such as insufficient consideration of automated system SCP needs and insufficient clarity to know how to best classify a method. Nonetheless, relationships can be established between our classification and others.

We consider that the classification can be useful for both researchers and practitioners. A more precise classification of V&V methods for SCP of automated systems can help others to better determine the circumstances under which a given method should be used, possible improvements and extensions on the methods, methods that can combined to jointly cover a wider V&V scope, and areas in which new methods could be needed.

The rest of the paper is organised as follows. Section 2 reviews related work. Sections 3 and 4 present the classification and its application, respectively. Section 5 discusses the classification. Finally, Section 6 summarises our main conclusions.

2 Related Work

As part of the work done in the VALU3S project to determine how to best classify V&V methods for SCP of automated systems, we searched for and analysed existing method classifications to analyse their adoption in the project. In this section, we review the main method classifications identified.

Nair et al. [Ref] identified evidence types for certification of safety-critical systems and created a taxonomy. Results of V&V methods was one of the evidence types. This type was refined into tool-supported V&V results and manual V&V results. The former was divided into testing results (13 basic types classified as objective-based testing, environment-based testing, or target-based testing), simulation, and formal verification results (three basic types). Similar V&V method types are referred to in engineering standards for safety-critical systems, e.g. EN 50128 [Ref] for railway software. The main issues with the classification by Nair et al. are that it focuses on safety, thus cybersecurity and privacy aspects are not sufficiently covered, and that it pays a much larger attention to testing than to other method types. This results in an unbalanced classification for our purpose.

The Amalthea project [Ref] worked on the development of an open-source tool platform for engineering embedded multi-and many-core software systems. To this end, V&V methods were reviewed and divided into Informal methods, Static methods, Dynamic methods, Formal methods, Testing, Simulation, and Product line analysis. The main issue that we found in this classification was that it was not clear how some methods should be classified, e.g., dynamic methods vs. formal methods or testing, as defined by the project. SCP requirements are also not explicitly addressed.

We identified the same issues mentioned above with several other classifications, e.g. one proposed by US Department of Defense [Ref]. This classification distinguishes four main method types: Informal V&V methods, Static V&V methods, Dynamic V&V methods, and Formal V&V methods. These classification types are commonly used. However, we consider that it is necessary to distinguish among informal, semi-formal, and formal methods, as well as explicitly among different types of dynamic methods such as testing and simulation because of their differences. This distinction is typical in engineering standards such as IEC 61508 [Ref], thus it is a relevant aspect for systems in regulated application domains.

There also exist classifications that specify the V&V methods that could be used in the different system lifecycle activities [Ref, Ref]. We regard these classifications in isolation as less useful because they do not represent well the reasons to use a method, how formal it is, or the type of requirements addressed.

3 Classification for V&V Methods for SCP of Automated Systems

This section presents the classification that we propose for V&V methods. It is the result of an effort in the VALU3S projects to decide upon how to best categorise V&V methods that we identified as relevant for evaluation of SCP of automated systems. We also analysed the methods [Ref]. The current structure of the classification is the result of several iterations and has been discussed among VALU3S partners.

The classification is based on two main facets of the V&V methods: the general method type and the concern addressed. When categorising a method, a user of the classification must choose (1) one or several general method types and (2) one or several concerns. This is justified in the next paragraphs.

The general method types considered are:

- **Injection**, when some phenomenon is introduced in a system to analyse its response.
- **Simulation**, when the behaviour of a model of a system is studied.
- **Testing**, when system execution under certain conditions is checked before operation.
- **Runtime verification**, when system execution is evaluated during operation.
- **Formal analysis**, for V&V methods with a mathematical basis.
- **Semi-formal analysis**, for V&V methods that exploit some structured means but without a full mathematical basis.
- **Informal analysis**, for V&V methods that do not follow any predefined structure or have mathematical basis.

We have identified five main concerns that SCP V&V methods for automated systems might have to address:

- **Safety**, as the ability of a system to avoid injury, serious injury, or death.
- **Cybersecurity**, as the ability of a system to avoid unauthorised access, use, or modification.
- **Privacy**, as the ability of a system to avoid disclosure of sensitive data.
- **General**, when a V&V method analyses a general characteristic of a system that does not directly contribute to SCP, but indirectly, e.g. traceability.
- **System-type-focused**, when a V&V method focuses on specific and distinguishing characteristics of a system type, e.g. a method for CPUs.

Among the characteristics that differentiate the classification and its use, we believe that considering injection as a separate independent general method type is very important for automated systems. Injection-based V&V methods focus on SCP evaluation, are essential for early system V&V, and can cope well with V&V of specific characteristics of cyber-physical systems, addressing injection from the software, hardware, network, mechanical, and real-world environment perspectives.

We also treat methods in a way that allows a user of the classification to consider very specific methods or broader ones. This is inspired by how engineering standards [Ref, Ref] present methods and it is also in line with how VALU3S industrial partner distinguish V&V methods. For example, the standards can refer both to general methods and categories such as performance testing and to specialisations such as stress testing and response time analysis. Therefore, the classification needs to be flexible regarding the abstraction level of the methods. This also implies that the classification of broader methods, for which specialised ones or sub-methods could be distinguished, might not be mapped to a single general method type and concern, but to several. For example, failure detection and diagnosis in robotic systems can be mapped to simulation and runtime monitoring as general method type and to safety and cybersecurity as concerns. This is shown in more detail in Section 4.

The following sub-sections present each general method type and how specific methods can be mapped to them, also considering the different concerns.

3.1 Injection

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: fault injection and attack injection.

Fault injection consists of the deliberate insertion of artificial (yet realistic) faults in a computer system or component in order to assess its behaviour in the presence of faults and allow the characterization of specific dependability measures and/or fault tolerant mechanisms available in the system. According to the well-known concepts and terminology proposed by Avizienis [Ref], a fault is the "adjudged or hypothesized cause of an error", and an "error is the part of the total state of the system that may lead to its subsequent service failure". In other words, the faults injected may lead to errors that, subsequently, may cause erroneous behaviour of the target component. These errors may propagate in the system and may cause failures in other components or even system failures. Fault injection can be seen as an approach to accelerate the occurrence of faults in order to help in the verification and validation of fault handling mechanisms available in the system under evaluation.

Avizienis et al. [Ref] define an attack as a special type of fault which is human made, deliberate and malicious, affecting (or breaching) hardware or software from external system boundaries and occurring during the operational phase. The system breach exploits the vulnerabilities in a system and could result into a compromised system. The compromised system could result in a system failure such as, software or hardware complete failure or degraded performance. Thus, attack injection in a system is analogous to fault injection. However, the aim is to evaluate the impact of cybersecurity attacks on the overall security of a system.

Fault and attack injection can be used in different phases of the systems development to evaluate (or even predict) how systems and specific components behave in the presence of faults, or to assess dependability properties such as safety, security, availability or reliability. Typically, faults injected in models (structural or behaviour-based models) are useful in the early stages of system development, while faults injected in prototypes or in real systems in controlled experiments allow the verification and validation of actual properties of deployed systems.

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Model-implemented fault injection* [Ref], to evaluate the safety aspects of the system's design by injecting fault models directly into simulated system models (such as MATLAB Simulink models) in early product development phases.
- **Cybersecurity and Privacy**: *Vulnerability and attack injection in real systems or prototypes* [Ref], to evaluate globally how the system copes with attacks and to assess specific security mechanisms in the target systems.
- **System-type-focused**: *Failure detection and diagnosis in robotic systems* [Ref], to analyse failures and possible failures in robotic system components via fault injection.

3.2 Simulation

Simulation based V&V methods are known for enabling early verification and validation of the systems and sub-systems. Simulation in basis is defined as process of developing or using digital models which behaves or operates like real world systems or components and providing real-like behaviour and outputs. Simulation based V&V methods provide virtual validation in software-intense systems. Risks are related to automated system such as faults, vulnerabilities, and threats can be experimented through simulation-based V&V methods.

Simulation-based V&V methods provide solutions for different challenges to efficiently enable early verification and validation. For example, simulation methods enable integration tests and behaviour tests without dealing expensive hardware or test equipment. Test scenarios can be created easier in most of real-world scenarios. Simulation based test solutions do not create risk for safety in the cases where human-machine interaction is existed. However, development of simulation and test processes is known as front loading process. So, the system complexity and efficiency of simulation-based V&V processes should be considered early.

Simulation based V&V, propose different approaches for tackling changes in V&V processes. an approach for the simulation-based V&V at early stages is coupling simulation models and simulators, existing code, and virtual hardware platforms. simulation-based fault injection is another approach for verification of robotic safety. Also, a proposed method several indexes on safety and efficiency of human-robot collaborations are considered. Another approach provide solution for Machine Learning-based systems which provides simulation environments for perception, planning and decision-making systems. Also, in collaborative tasks, simulation-based systems enable V&V of safety-critical systems via virtual & augmented reality technologies.

As an overview of V&V methods, Simulation is related to many other V&V methods like simulation-based attack injection in security, simulation-based fault injection in safety. Moreover, simulation-based systems are parts of Semi-formal methods, System type focused and testing methods. Simulation-based robot verification has fault injection mechanism to test robot trajectory in case of safety in faulty situations. Simulation related methods provide V&V without producing any physical item and adding risk to the environment and environment can provide powerful monitoring. On the other hand, simulation-based applications mostly run-on hierarchical models. This narrows availability of both academic and industrial resources in development and Simulation tools can require much computational power and limit real-time applications.

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Simulation-based robot verification* [ref], to assure a robot's trajectory safety and in turn to increase flexibility and robustness by maintaining the level of productivity.
- **Cybersecurity and Privacy**: *V&V of machine learning-based systems using simulators* [ref], which aims to provide efficient and effective V&V of SCP requirements in simulated environments without endangering human safety.

- **General**: *Virtual and augmented reality-based user interaction V&V* [ref], for human factors analysis and technology acceptance by end users using virtual or augmented reality technologies before the system is fully deployed.
- **System-type-focused**: *CPU verification* [Ref], to ensure that a CPU delivers its functionality correctly and as intended, and which can exploit simulation to this end.

3.3 Testing

This group of methods focuses on validating a system by its execution 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 perform 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. This environment is sometimes called test harness. Further, usually a means is needed to judge whether the SUT's reactions conform to expectations. Such means is sometimes referred to as test oracle. For testing, the SUT can be the final system as well as any artefact used during its development; i.e., models or specific hardware or software components.

It is distinguished between black box testing – where only the interfaces of the SUT are considered, and its interior considered as black box; white box testing – where also the SUT's interior, e.g. inner states, is monitored; and combinations of both, i.e. grey box testing. The scope of testing can be functional, i.e. assessing whether the SUT behaves as expected (fulfils its functions), and non-functional, i.e. assessing its performance, robustness, security etc. Therefore, testing can contribute significantly to establish SCP. However, it should be considered that testing is usually incomplete, i.e. even successful passing a large set of test cases (a test suite) is no guarantee for the SUT's correctness. A test suite's quality is correlated with two aspects: how good it covers the addressed issues (functionality, robustness, …), and how efficiently it achieves this.

A technique to get high quality test cases is (automated) test case generation, which is used by most of the methods described in this clause. Further, various coverage criteria are addressed, e.g. scenarios, potential implementation faults, or potential impact of cybersecurity attacks on safety. Many of the V&V-methods address testing of non-functional issues such as safety, robustness, cyber-security, but also with novel properties of automated systems, e.g. machine learning. Some reviewed methods can also be used for functional testing. All types of components are considered, with a slight emphasis on models, in order to detect conceptual flaws as early as possible. Finally, black-box and white-box testing are supported.

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Model-Based Robustness Testing* [Ref], to derive unexpected or slightly out of specification stimuli in order to check the robustness of the system or component under test.
- **Cybersecurity**: *Assessment of cybersecurity-informed safety* [Ref], to black-box test security-informed safety of automated driving systems and in turn produce an understanding of the interplay between safety and security.

- **Privacy**: TBD
- **General**: *Model-based testing* [Ref], to derive tests from (semi-)formal behaviour models or test models.
- **System-type-focused**: *Penetration testing of industrial systems* [Ref], to analyse sensor data and server-PLC communication for evaluation of system robustness in the case of sensor data manipulation and of effects of data manipulation in communication.

3.4 Runtime Verification

Runtime Verification denotes a subset of formal methods that trade the computationally costly approach adopted by exhaustive offline verification techniques by a lightweight and limited, but still rigorous and precise, runtime kind of verification [ref]. RV methods are usually applied to verify properties that are either impossible or computationally impractical to be verified statically with the SOTA exhaustive formal verification approaches [ref].

This section focuses on RV methods that use monitors to verify, during runtime, that a system's behavior correctly complies with its formal specification. In this context, behavior expresses how the system evolves concerning time and its states. To issue such verdicts, monitors collect and analyse data in the form of traces, using it to verify if the current system state, or a set of recorded system actions, comply with a given specification. Such formal specifications are typically described in various flavors of temporal logics, state machines, and regular expressions.

Although RV solutions have a broad spectrum of applicability, embedded-safety critical systems seem to be the research field where it shines the most. Considering the high level of safety and security required by such systems, RV is becoming widespread given its ability to identify faulty behavior accurately and in a timely way, given its lightweight resource usage.

given the complexity associated with the development of autonomous driving systems, exhaustive formal verification techniques commonly run into state explosion problems. As an alternative, RV techniques have been employed to analyse safety requirements of system components like Adaptive Cruise Controls and their respective various PID control parameters by using Signal Temporal Logic

RV has been applied as a key technique against memory cyber-attacks. It consists of a continuous analysis and verification of a control flow graph that represents the target system to guarantee that no illegal transition happens between instruction segments (e.g., the program being erroneously/maliciously redirected to an undesired memory address).

the European General Data Protection Regulation (GDPR) describes guidelines on how citizens' data should be handled by organizations. RV solutions have been used to identify violations of privacy and to evidentiate that systems have been complying with the guidelines.

analog/mixed-signal systems models have been using RV solutions to guarantee their functional correctness instead of exhaustive simulation-based approaches given the reduced time to verify and validate that it offers

RV solutions have been applied in the real-time domain to verify if the task activation patterns of a real-time system complies a formal model responsible for describing task-set time upper bounds

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Dynamic analysis of concurrent programs* [Ref], to find errors in synchronisation of concurrently executing threads, processes, or any other tasks executed concurrently.
- **Cybersecurity and Privacy**: *Test oracle observation at runtime* [Ref], to dynamically assess the robustness of system behaviour during its runtime by measuring how far the system is from satisfying or violating a property expressed in a formal specification language.
- **General**: *Runtime verification based on formal specification* [Ref], to formally specify properties of runtime observations and verify them using automatically generated monitors.
- **System-type-focused**: *Model-based formal specification and verification of robotic systems* [Ref], to enable formal verification of robotic systems by developing models that cope with the intractable state space of complex robotic system software, improving the verification coverage and assurance by combining formal methods and runtime verification.

3.5 Formal Verification

Formal Verification denotes a set of methods intended to prove properties of a system with formal methods based on mathematical models of the system. Compared to previous methods, Formal Verification is not focused on single executions of the system, but on proving properties exhaustively on all executions based on some mathematical models. Although the name refers to verification only, it comprises both validation and verification methods: for verification, the properties formalize the system requirements specification, while for validation, the properties are used to check if the model is the right representation of the system (e.g., consistency checking, reachability of states, vacuous satisfaction of requirements).

Model checking [MCH1, MCH2, MCH3] represents a prominent class of Formal Verification methods. Model checking uses a variety of languages to write the system models, which range from finite-state to infinite-state machines, from discrete-time to timed or hybrid systems, from non-deterministic automata to stochastic models, from synchronous to asynchronous communicating programs. Given a formal semantics of the input language, model checking can also be applied to models defined for other purposes (architectural description or simulation) or directly to software or hardware source code. Also, for the property specification, there is a wide range of options, ranging from simple reachability or invariant properties, to temporal properties, from safety to liveness properties. Depending on the modelling language, temporal properties can be specified in various logics, either propositional or first-order, discrete or continuous or hybrid time, linear or branching or probabilistic or hyper-properties logic. The model checking problem is solved algorithmically by a procedure that decides if the model satisfies the property or finds a counterexample that shows how the model violates it. When the problem is undecidable (as for example for software), the model checking procedure may be incomplete.

Another major class of Formal Verification methods is based on Deductive verification. Properties and systems are usually represented in first-order logic, higher-order logics, or specific theories (arithmetic, sets, continuous functions) to allow modelling specific aspects of CPS. Deductive verification methods are based on the generation of proof obligations that encode the correctness of the system. Depending on the underlying logic, these proof obligations are discharged by interactive theorem provers (such as HOL, ACL2, Isabelle, or Coq), automatic theorem provers where the proof is extracted from the specification and additional annotations, and Satisfiability Modulo Theories (SMT) solvers (Z3, Yices, MathSAT5, Alt-Ergo, CVC3).

In formal verification, “safety” refers to a class of properties that require some bad condition will not happen. In this sense, most formal verification techniques manage “safety” properties. However, here we refer to safety in the context of dependability as the absence of catastrophic consequence on user and environment. Safety is thus strictly connected to reliability and the ability of the system to continue functioning in the presence of faults. In this context, examples of formal verification for safety include model-based safety analysis techniques, which perform minimal cut set analysis to analyze the failure of the system in different combinations of faults.

Examples of formal verification of cybersecurity include verification of cybersecurity protocols, which require modeling of attackers and the knowledge they gather from interacting with the protocols.

It is sometimes hard to distinguish from cybersecurity, but we can list here formal verification techniques focused on data protection and thus information flow analysis.

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Formal requirements validation* [Ref], to confirm the validity of the specification of formal requirements in terms of consistency, compatibility with scenarios, vacuity, realizability, and other formal checks that contribute to system safety.
- **Cybersecurity and Privacy**: *Source code static analysis* [Ref], to derive various runtime properties and find various kinds of errors in programs without executing them at all or at least not under their original semantics, and which can address cybersecurity and privacy considerations.
- **General**: *Source code static analysis* [Ref], to derive various runtime properties and find various kinds of errors in programs without executing them at all or at least not under their original semantics.
- **System-type-focused**: *Reachability analysis-based verification for safety-critical hybrid systems* [Ref], to exhaustively explore a system’s evolution over time, given an initial input range.

3.6 Semi-Formal Analysis

This method type deals with system evaluation by using structured means whose application does not result in a mathematical proof. The methods enable that confidence in system dependability is developed in relation to characteristics of an automated system such as faults, vulnerabilities, and threats. The methods also contribute to the avoidance and identification of these issues, and the recovery from them.

As a mathematically rigorous approach to the SCP V&V of complex systems is unfeasible in many cases, semi-formal techniques are used to complement formal V&V. System decomposition, abstraction, and specific models reduce SCP V&V to sub-problems of limited scope that may be addressed using semi-formal methods and tools, which can rely on models, architectural principles, mathematical or probabilistic calculus, qualitative and quantitative analysis, and simulation, while addressing engineering and assurance standards [refs].

Semi-formal analysis also enables the evaluation of general characteristics of a system that contribute to SCP, e.g. about the traceability between system artefacts. These characteristics indirectly address automated system SCP by confirming the fulfilment of aspects that contribute to it. For instance, requirements traceability contributes to assuring that the correct and expected functionality has been implemented in a system. This in turn contributes to developing confidence in system reliability and consequently in SCP. In other words, if someone cannot confirm that the correct and expected functionality has been implemented in a system, it might not be possible to develop sufficient confidence in system SCP.

Examples of V&V methods of this type for the different concerns include:

- **Safety**: *Model-based safety analysis* [ref], 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 much of the safety analysis.
- **Cybersecurity**: *Wireless interface network security assessment* [Ref], which aims to analyse a system's robustness against network security attacks carried out through wireless interfaces by evaluating CANBUS-based control network security and teleoperation and supervision network security.
- **Privacy**: *Model-based assurance and certification* [Ref], to justify system dependability in compliance with privacy standards.
- **General**: *Knowledge-centric system artefact quality analysis* [Ref], to quantitatively determine the suitability of system artefacts by exploiting ontologies and semantic information, and according to selected criteria such as correctness, consistency, and completeness.
- **System-type-focused**: *Model-based avionics software specification and verification* [Ref], based on the modelling of the DO-178C standard.

3.7 Informal Analysis

Although in VALU3S we have not reviewed informal analysis methods, we include them in our classification for completeness. These methods are based on human reasoning and subjectivity, without a predefined underlying formalism or structure. Nonetheless, they are used in industry.

Walkthroughs are among the most common informal analysis methods. They correspond to the situation in which the producer of some system artefact presents the artefact to others for defect identification. A programmer performing a source code peer review is another example. In both cases, the application of the method could aim to detect SCP issues in a system or some system artefact, as well as target the analysis of some general or system-type-focused characteristic.

4 Application of the Classification

The proposed classification scheme has been defined by the joint effort of more than 70 people who hold various positions, such as researchers, system engineering, and tool vendors, and the needs of 31 different entities, including big companies, small and medium-sized enterprises, and universities working in different fields.

To give precise guidelines about how to apply the proposed classification, we consider a set of 53 reference methods for automated systems. This set contains commonly-used methods as well as methods that will be improved and new methods that will be created by combination of methods. The complete list of these methods with the related classification category and addressed concerns (Safety – Sa, Cybersecurity – C, Privacy – P, General – G, System-type-focused – Sy) is reported in the following:

1. **Injection**: Fault Injection in FPGAs (Sa), Model-Implemented Fault Injection (Sa), Simulation-based fault injection at system-level (Sa), Software-implemented fault injection (Sa, G), Model-based fault injection for safety analysis (Sa, Sy), Model-Implemented Attack Injection (C), Simulation-based attack injection at system-level (C), Vulnerability and attack injection (C), Interface fault injection (G).
2. **Simulation**: Simulation-based testing for human-robot collaboration (Sa), V&V of machine learning-based systems using simulators (Sa, C), Virtual & augmented reality-based user interaction V&V and technology acceptance (Sa, G), Simulation-based robot verification (Sa, Sy), Test optimization for simulation-based testing of automated systems (Sa, Sy), Virtual architecture development and simulated evaluation of software concepts (Sa, Sy).
3. **Testing**: Software component testing (Sa), Assessment of cybersecurity-informed safety (Sa, C), Machine learning model validation (Sa, Sy), Behaviour-driven model development and test-driven model review (Sy), Model-based mutation testing (Sy), Model-based robustness testing (Sy), Model-based testing (Sy), Risk-based testing (Sy), Signal analysis and probing (Sy), System-type-focused (Sy), Test parallelization and automation (Sy).
4. **Runtime verification**: Dynamic analysis of concurrent programs (Sa, Sy), Runtime verification based on formal specification (Sa, Sy), Test oracle observation at runtime (Sa, Sy).
5. **Formal Analysis**: Deductive verification (Sa, Sy), Behaviour-driven formal model development (Sa, Sy), Formal requirements validation (Sa, Sy), Model checking (Sa, Sy), Reachability-analysis-based verification for safety-critical hybrid systems (Sa, Sy), Theorem proving and Satisfiability modulo theories solving (Sa, Sy), Source code static analysis (Sa, C, P, G, Sy).
6. **Semi-formal Analysis**: Human interaction safety analysis (Sa), Traceability management of safety software (Sa), Code design and coding standard compliance

checking (Sa), Risk analysis (Sa, C), Model-based safety analysis (Sa, C, Sy), Knowledge-centric system artefact quality analysis (Sa, G), Model-based design verification (Sa, Sy), Intrusion detection for wireless sensor networks based on Weak Model Processes state estimation (C), Kalman filter-based fault detector (C), Model-based threat analysis (C), Vulnerability analysis of cryptographic modules against hardware-based attacks (C), Wireless interface network security assessment (C), Knowledge-centric traceability management (Sa, C, P, G, Sy), Model-based assurance and certification (Sa, C, P, G, Sy).

7. **Informal Analysis**: Model-based formal specification and verification of robotic systems (Sa, Sy), Failure detection and diagnosis in robotic systems (Sa, C, G), Central Processing Unit verification (Sa, C, G, Sy), Penetration testing of industrial systems (C).

5 Discussion

TBD

TBD

6 Conclusion

TBD

TBD

Acknowledgments. The research leading to this paper has received funding from the VALU3S (H2020-ECSEL grant agreement no 876852; Spain's MICINN ref. PCI2020-112001), iRel4.0 (H2020-ECSEL grant agreement no 876659; MICINN ref. PCI2020-112240), and Treasure (JCCM ref. SBPLY/19/180501/000270; European Regional Development Fund) projects, and from the Ramon y Cajal Program (MICINN RYC-2017-22836; European Social Fund). We are also grateful to all the VALU3S partners that have provided input and feedback for the development of the classification.

References

1. TBD