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

[Authors - Removed for double-blind review]

[Affiliations - Removed for double-blind review]

`[emails - Removed for double-blind review]`

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.

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 [4]. 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 [31]. In this paper we focus on methods for verification and validation (V&V) of automated systems. Examples of these methods are fault injection [30] and model-based testing [26].

The features of the new generation of automated systems require that dedicated V&V methods (usually a combination of methods) are applied to them [4,12]. 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 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 [XXX][1] [4], a large-scale industry-academia project in which [XXX] partners from [XXX] 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 [44] 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 [XXX] 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 be 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. Finally, Section 5 summarises our main conclusions.

[1] Label used to hide information for double-blind review

2 Related Work

As part of the work done in the [XXX] project to determine how to best classify V&V methods for SCP of automated systems, we searched for and analysed existing method classifications to assess their adoption in the project.

Nair et al. [29] identified evidence types for certification of safety-critical systems and created a taxonomy. Results of V&V methods is one of the evidence types. It is refined into tool-supported V&V and manual V&V methods. The former is divided into testing (13 basic types classified as objective-based testing, environment-based testing, or target-based testing), simulation, and formal verification (three basic types). Similar V&V method types are referred to in engineering standards for safety-critical systems, e.g., EN 50128 [9] 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 Amalthea4public project [1] 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 with several other classifications, e.g., one proposed by US Department of Defense [42]. 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 method types are commonly used in classifications. 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 [20], 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 [22]. 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.

In summary, our classification fills gaps in prior work by considering a broader range of general V&V method types, explicitly focusing on SCP, and providing a detailed description of its elements, how they can be distinguished, and how to use them.

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 [XXX] project 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 [43]. The current structure of the classification is the result of several iterations and has been discussed among [XXX] 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 do not have a 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 for critical systems [9,20] present methods and it is also in line with how [XXX] industrial partners 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 or 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

Injection-based V&V methods focus 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 specific types of injection are discussed: fault injection and attack injection.

Fault injection consists in the deliberate insertion of artificial (yet realistic) faults in a computer system or component. This way, it is possible to assess the behaviour of a system in the presence of faults and allow the characterization of specific dependability measures or fault tolerant mechanisms available in the system. According to the well-known concepts and terminology proposed by Avizienis et al. [3], 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 V&V of the fault handling mechanisms available in the system under evaluation.

Avizienis et al. [3] 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 system 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, privacy, 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 V&V of actual properties of deployed systems.

Examples of injection-based V&V methods for the different concerns include:

- **Safety**: *Model-implemented fault injection* [37], to evaluate the safety aspects of a system's design by injecting fault models directly into simulated system models (such as Simulink ones) at early product development phases.
- **Cybersecurity**: *Vulnerability and attack injection in real systems or prototypes* [15], to evaluate globally how a system copes with attacks and to assess specific security mechanisms in the target systems.
- **Privacy**: *SQL injection* [17], to assess the possibility of unrestricted access to databases.
- **General**: *Noise injection* [27], to add irrelevant data to the inputs during neural network training and assess the impact for V&V.
- **System-type-focused**: *Failure detection and diagnosis in robotic systems* [24], to analyse failures and possible failures in robotic system components via fault injection.

3.2 Simulation

Simulation enables early V&V of systems and components. It is based on the development or use of digital models that behave or operate like real-world systems or components, and on the provision of real-world-like outputs. Simulation-based V&V methods provide virtual validation in software-intensive systems. Possible issues in automated systems can be experimented and analysed through simulation.

This type of V&V methods provides solutions for different challenges for efficient early V&V. For example, simulation methods enable integration tests and behaviour tests without dealing with expensive hardware or test equipment. Test scenarios can be created easily for most of real-world scenarios. Simulation-based test approaches do not introduce direct safety risks in cases where human-machine interaction exists. However, the effort and cost of the development of simulation and its test processes can be high. The trade-off between simulation accuracy on the one hand, and simulation speed, resource consumption, and effort for constructing simulation models on the other hand, has to be considered.

Simulation supports different approaches for tackling challenges in V&V processes. An approach is the virtual validation of complex systems and system architectures by coupling simulation models and simulators, existing code, and virtual hardware platforms [25]. Another is the study of the safety and efficiency of human-robot collaboration [40]. There exist also approaches that provide solutions for machine learning-based systems through the provision of simulation environments for perception, planning, and decision-making systems [19].

Simulation is closely related to other V&V method types. Injection-based methods can be used in simulated environment. When aiming to highlight these characteristics, they can be referred to, e.g., as simulation-based attack injection for cybersecurity and simulation-based fault injection for safety. Simulation methods also usually exploit semi-formal methods such as models and testing aspects such as test case management.

A major advantage of simulation is that V&V can be conducted without producing any physical item and adding risk to the environment. On the other hand, simulation-based applications mostly run on hierarchical models. This narrows the availability of both academic and industrial resources in development. Simulation tools can require significant computational power and limit real-time applications.

Examples of simulation methods for the different concerns include:

- **Safety**: *Simulation-based robot verification* [40], to assure a robot's trajectory safety and in turn to increase flexibility and robustness by maintaining the level of productivity.
- **Cybersecurity**: *V&V of machine learning-based systems using simulators* [19], which aims to provide efficient and effective V&V of SCP requirements of machine learning in simulated environments without endangering human safety.
- **Privacy**: *Simulation of obfuscation and negotiation* [14], to safeguard location information.
- **General**: *Virtual and augmented reality-based user interaction V&V* [6], for human factor analysis and technology acceptance by end users before a system is built or deployed.
- **System-type-focused**: *CPU verification* [18], to ensure that a CPU delivers its functionality correctly and as intended, and which can exploit simulation.

3.3 Testing

This type of V&V methods focuses on validating a system by executing it in the frame of so-called test cases. A test case contains at least 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. To run a test case, an environment is used. It allows the tester to feed the SUT with the input data in a controlled manner, as well as to monitor SUT reactions. This environment is sometimes called test harness. Furthermore, usually a means is needed to judge whether SUT 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, such as models or specific hardware or software components. The methods that focus on testing of models are especially useful for early detection of conceptual flaws.

Among the different ways to distinguish them, testing approaches can be divided into black-box testing and white-box testing. In black-box testing, only the interfaces of the SUT are considered and its interior is considered as a black box. White-box testing monitors the SUT's interior, e.g., inner states. A combination of both, i.e., grey-box testing, is also possible. The scope of testing can be functional, when assessing whether the SUT behaves as expected (i.e., it fulfils its functions), and non-functional, when characteristics such as performance, robustness, and security are assessed. Therefore, testing can contribute significantly to establishing SCP. However, it must be considered that testing is usually incomplete. Even successfully 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 way to get high quality test cases is (automated) test case generation, which is used by many testing methods. Furthermore, various coverage criteria can be addressed, such as scenarios, potential implementation faults, or potential impact of cybersecurity attacks on safety. Many V&V methods for critical systems address testing of non-functional issues such as safety, robustness, and cybersecurity, and also novel properties of automated systems, e.g., machine learning.

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

- **Safety**: *Model-Based Robustness Testing* [38], 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* [39], 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**: *System testing for GDPR compliance* [33], to confirm adherence to GDPR requirements before a system is deployed.
- **General**: *Model-based testing* [26], to derive tests from (semi-)formal behaviour models or to test models.
- **System-type-focused**: *Penetration testing of industrial systems* [44], to analyse sensor data and server-PLC communication for evaluation (1) of system robustness in the case of sensor data manipulation and (2) of effects of data manipulation in communication.

3.4 Runtime Verification

Runtime verification is a method that allows checking whether a run of a target system satisfies or violates a set of correctness properties. It trades the computationally costly approach adopted by exhaustive offline verification techniques by a lightweight and limited, but still rigorous and precise, verification mechanism during execution time.

This V&V method type uses monitors to verify that a system's behaviour complies with its specification. In this context, behaviour expresses how the system evolves concerning the passage of time and its states' changes. To issue such verdicts, monitors collect and analyse execution traces, using them to verify if the current system state, or a set of recorded system actions, complies with a given specification. Such a specification is in general encoded in some formal language typically belonging to the family of temporal logics, state machines, rule systems, or regular expressions.

The process of collecting data from the system and feeding them to monitors, called instrumentation, is an essential part of this method type. Ideally, the instrumentation process should be considered at design time, as overheads can be minimized and performance can be optimized. However, legacy systems could also benefit from runtime verification solutions. The way monitors can be implemented is classified in various ways [8], covering aspects such as temporal and logical isolation from the monitored system, how much a monitor synchronizes with the system's execution flow, and whether a monitor is hardware- or software-based.

Although runtime verification solutions have a broad spectrum of applicability, V&V of embedded safety-critical systems seems to be the area where it shines the most. Considering their demanding safety and security levels, runtime verification is becoming widespread given its ability to identify faulty behaviour accurately and in a timely way, given its formal reasoning and lightweight resource usage.

Runtime verification methods are also especially suitable to verify properties that static formal verification techniques fail to confirm in a timely and resource-constrained way. On top of that, runtime verification tools need fewer model assumptions to work, which is also a notorious downside of static tools. Testing is another area that lately has been benefiting from runtime verification solutions as it can complement traditional testing techniques, speeding up the validation of complex system parts.

Examples of runtime verification methods for the different concerns include:

- **Safety**: *Dynamic analysis of concurrent programs* [13], to find errors in synchronisation of concurrently executing threads, processes, or any other tasks executed concurrently.
- **Cybersecurity**: *Test oracle observation at runtime* [5], to dynamically assess system behaviour by measuring how far the system is from satisfying or violating a property specified formally, e.g., a cybersecurity property.
- **Privacy**: *Monitoring of GDPR compliance* [2], to confirm adherence to GDPR requirements after a system is deployed, and to identify violations.
- **General**: *Runtime verification based on formal specification* [10], to formally specify properties of runtime observations and verify them with monitors.
- **System-type-focused**: *Model-based formal specification and verification of robotic systems* [28], to formally verify these systems with models that cope with the intractable state space of complex systems, improving V&V coverage and assurance by combining formal methods and runtime verification.

3.5 Formal Analysis

Formal analysis denotes a set of methods to prove properties of a system with formal methods based on mathematical system models. Formal analysis is typically not focused on single executions of a system, but on proving properties exhaustively on all executions. Formal analysis comprises both V&V: 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, and vacuous satisfaction of requirements.

Model checking [11] is a prominent class of formal analysis methods. It uses a variety of languages to represent systems, from finite-state to infinite-state machines, from discrete-time to timed or hybrid systems, from non-deterministic automata to stochastic models, and 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, e.g., architectural description or simulation, or directly to source code. There is a wide range of options for property specification, ranging from simple reachability or invariant properties to temporal properties, and from safety to liveness properties. Depending on the modelling language, temporal properties can be specified in different logics, e.g., either propositional or first-order. The model checking problem is solved algorithmically by a procedure that decides if the model satisfies the property or finds a counterexample. When the problem is undecidable, e.g., for software, the model checking procedure may be incomplete.

Another major class of formal analysis methods corresponds to those based on deductive verification [11]. Properties and systems are usually represented in first-order logic, higher-order logics, or specific theories (arithmetic, sets, continuous functions). 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, automatic theorem provers where the proof is extracted from the specification and additional annotations, or Satisfiability Modulo Theories (SMT) solvers.

In general, formal analysis methods are independent of specific concerns and can be applied to SCP concerns as long as the requirements can be formalised in the property language. There exist however more specific formal analysis methods for SCP taking into account faults or attacker models.

Examples of formal analysis methods for the different concerns include:

- **Safety**: *Model-based safety analysis* [7], to formally analyse the fault configurations leading to a system failure, given a behavioural model.
- **Cybersecurity**: *Source code static analysis* [36], to derive various runtime properties and find various kinds of errors in programs without executing them, and which can address cybersecurity considerations.
- **Privacy**: *GDPR compliance formal verification* [23], to formally ensure that a system satisfies GDPR requirements.
- **General**: *Deductive verification* [21], to ensure that source code conforms to its formal specification.
- **System-type-focused**: *Reachability analysis-based verification for safety-critical hybrid systems* [41], to exhaustively explore a system's evolution over time, given an initial input range.

3.6 Semi-Formal Analysis

This type of V&V methods deals with the evaluation of systems and components 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 risks, faults, vulnerabilities, and threats. The methods also contribute to the avoidance and identification of these issues, and to the recovery from them.

As a mathematically rigorous approach to 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. These methods and tools can rely on models, architectural principles, mathematical or probabilistic calculus, qualitative and quantitative analysis, and simulation, among other means, while also addressing compliance with engineering and assurance standards.

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 conditions needed for SCP. 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 dependability assessment* [16], with which system and safety engineers share a common system model created using a model-based development process and extend the system model with a fault model as well as relevant portions of the physical system to be controlled, also enabling safety analysis automation.
- **Cybersecurity**: *Wireless interface network security assessment* [32], to analyse a system's robustness against network security attacks carried out through wireless interfaces by evaluating (1) CANBUS-based control network security and teleoperation and (2) supervision network security.
- **Privacy**: *Model-based assurance and certification* [12], to justify system dependability in compliance with standards, e.g., privacy ones, by taking advantage of structured information specifications about a system, about the standards, and about their relation.
- **General**: *Knowledge-centric system artefact quality analysis* [34], 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* [35], which is based on the modelling of the DO-178C standard and can contribute to requirements V&V, among other tasks.

3.7 Informal Analysis

Although in [XXX] 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.

Walkthrough [20] is among the most common informal analysis methods. It corresponds to the situation in which the producer of some system artefact presents it 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, as well as to analyse some general or system-type-focused characteristic.

4 Application of the Classification

The proposed classification scheme has been used for the review of state-of-the-art and state-of-the-practice V&V methods in the [XXX] project [43]. Such usage shows how the classification has helped in a real V&V method classification effort. The usage outcome also allows us to claim that the classification can be a feasible means in practice for the classification of V&V methods for SCP of automated system.

Seventy-three people from 31 organisations contributed to the review of 53 V&V methods for automated systems. The people cover different roles for automated system V&V, such as researchers, systems engineers, and tool vendors, and the organisations include large enterprises, small and medium-sized enterprises, and research institutions from the automotive, agriculture, railway, healthcare, aerospace, and industrial automation domains. The complete list of methods is presented in Table 1, considering their different method types and concerns (Safety – Sa, Cybersecurity – C, Privacy – P, General – G, System-type-focused – Sy). Some V&V methods map to several types. For instance, simulation-based robot verification uses both simulation and runtime verification. Further information about the review of the methods can be found in [43].

5 Conclusion

It is essential that the manufacturers and component suppliers of automated systems use adequate verification and validation (V&V) methods to confirm that the systems' safety, cybersecurity, and privacy (SCP) requirements are satisfied. This requires that the manufacturers and suppliers clearly understand the characteristics of the methods, when the methods should be used, and for what purposes.

We have presented a new classification scheme to categorise V&V methods used to evaluate automated systems with respect to SCP requirements. The scheme provides practitioners and researchers with a clear and easy-to-understand set of categories where V&V methods could be selected from. The method types considered are injection, simulation, testing, runtime verification, formal analysis, semi-formal analysis, and informal analysis. The methods can deal with different concerns: SCP, general concerns that indirectly contribute to SCP, or system-type-focused concerns.

The scheme has been successfully used by 73 researchers and practitioners to classify 53 V&V methods, covering six different application domains. This makes us confident in the validity of the classification scheme.

Table 1. Classification of V&V methods with the proposed scheme.

Injection: Fault injection in FPGAs (Sa, Sy), Interface fault injection (G), Model-based fault injection for safety analysis (Sa), Model-implemented attack injection (C), Model-implemented fault injection (Sa), Simulation-based attack injection at system-level (C), Simulation-based fault injection at system-level (Sa), Software-implemented fault injection (Sa, G), Vulnerability and attack injection (C).
Simulation: Assessment of cybersecurity-informed safety (Sa, C), CPU verification (Sa, C, G, Sy), Failure detection and diagnosis in robotic systems (Sa, C, G, Sy), Fault injection in FPGAs (Sa, Sy), Kalman filter-based fault detector (C), Model-implemented attack injection (C), Model-implemented fault injection (Sa), Simulation-based fault injection at system-level (Sa), Simulation-based fault injection at system-level (Sa), Simulation-based testing for human-robot collaboration (Sa, Sy), Test optimization for simulation-based testing of automated systems (Sa, G), 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, G, Sy), Virtual architecture development and simulated evaluation of software concepts (Sa, G).
Testing: Assessment of cybersecurity-informed safety (Sa, C), Behaviour-driven formal model development (Sa, G), Behaviour-driven model dev. and test-driven model review (G), CPU verification (Sa, C, G, Sy), Fault injection in FPGAs (Sa, Sy), Interface fault injection (G), Intrusion detection for wireless sensor networks based on Weak Model Processes state estimation (C), Machine learning model validation (Sa, G, Sy), Model-based mutation testing (G), Model-based robustness testing (G), Model-based testing (G), Penetration testing of industrial systems (C, Sy), Risk-based testing (G), Signal analysis and probing (G, Sy), Simulation-based testing for human-robot collab. (Sa, Sy), Software component testing (Sa, G), Software-implemented fault injection (Sa, G), Test parallelization and automation (G), Vulnerability and attack injection (C), Wireless interface network security assessment (C).
Runtime verification: Behaviour-driven model development and test-driven model review (G), Dynamic analysis of concurrent programs (Sa, G, Sy), Failure detection and diagnosis in robotic systems (Sa, C, G, Sy), Fault injection in FPGAs (Sa, Sy), Model-based formal specification and verification of robotic systems (Sa, G, Sy), Runtime verification based on formal specification (Sa, G), Simulation-based robot verification (Sa, G, Sy), Test oracle observation at runtime (Sa, G).
Formal Analysis: Behaviour-driven formal model development (Sa, G), CPU verification (Sa, C, G, Sy), Deductive verification (Sa, G), Formal requirements validation (Sa, G), Model checking (Sa, G), Model-based design verification (Sa, G), Model-based fault injection for safety analysis (Sa), Model-based formal specification and verification of robotic systems (Sa, G, Sy), Model-based safety analysis (Sa, C), Reachability-analysis-based verification for safety-critical hybrid systems (Sa, G, Sy), Source code static analysis (Sa, C, P, G), Theorem proving and satisfiability modulo theories solving (Sa, G), V&V of machine learning-based systems using simulators (Sa, C).
Semi-formal Analysis: Behaviour-driven model development and test-driven model review (G), Code design and coding standard compliance checking (Sa), Failure detection and diagnosis in robotic systems (Sa, C, G, Sy), Human interaction safety analysis (Sa), Intrusion detection for wireless sensor networks based on Weak Model Processes state estimation (C), Kalman filter-based fault detector (C), Knowledge-centric system artefact quality analysis (Sa, C, P, G), Knowledge-centric traceability management (Sa, C, P, G), Model-based assurance and certification (Sa, C, P, G), Model-based design verification (Sa, G), Model-based safety analysis (Sa, C), Model-based threat analysis (C), Risk analysis (Sa, C), Source code static analysis (Sa, C, P, G), Traceability management of safety software (Sa), Vulnerability analysis of cryptographic modules against hardware-based attacks (C), Wireless interface network security assessment (C).

As future work, we will continue classifying V&V methods with the proposed scheme, e.g., methods used in other domains and methods combined or developed in the [XXX] project. This will allow us to further validate the classification.

Acknowledgments. [Removed for double-blind review]

References

1. Amalthea4public project: D3.1 - Analysis of state of the art V&V techniques. 2015
2. Arfelt, E., et al.: Monitoring the GDPR. ESORICS 2019.
3. Avizienis, A., et al.: Fundamental Concepts of Dependability. University of Newcastle, Computing Science, 2001
4. [Removed for double-blind review]
5. Bartocci, E., et al.: Automatic Failure Explanation in CPS Models. SEFM 2019
6. Belmonte, L., et al.: Feeling of Safety and Comfort Towards a Socially Assistive Unmanned Aerial Vehicle That Monitors People in a Virtual Home. Sensors 21(3): 908, 2021
7. Bozzano, M., et al.: Efficient Anytime Techniques for Model-Based Safety Analysis. CAV 2015
8. Cassar, I., et al: A Survey of Runtime Monitoring Instrumentation Techniques. PrePost@iFM 2017
9. CENELEC: EN 50128 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. 2020
10. Cimatti, A., et al.: Assumption-Based Runtime Verification with Partial Observability and Resets. RV 2019
11. Clarke, E.M., et al.: Handbook of Model Checking. Springer, 2018
12. de la Vara, et al.: Assurance and Certification of Cyber-Physical Systems: The AMASS Open Source Ecosystem. Journal of Systems and Software 171: 110812, 2021
13. Dias, R., et al.: Verifying Concurrent Programs Using Contracts. ICST 2017.
14. Duckham, M., Kulik, L.: Simulation of Obfuscation and Negotiation for Location Privacy. COSIT 2005
15. Fonseca, J., et al.: Analysis of Field Data on Web Security Vulnerabilities. IEEE Transactions on Dependable and Secure Computing 11(2): 89-100, 2014
16. Gallina, B., et al.: Multi-concern Dependability-centered Assurance for Space Systems via ConcertoFLA. Ada-Europe 2018
17. Halfind, W.G.J., et al.: A Classification of SQL Injection Attacks and Countermeasures. ISSSE 2006
18. Herdt, V., et al.: Efficient Cross-Level Testing for Processor Verification: A RISC- V Case-Study. FDL 2020
19. Humbatova, N., et al.: Taxonomy of real faults in deep learning systems. ICSE 2020
20. IEC: IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems. 2011
21. Hähnle R., Huisman M.: Deductive Software Verification: From Pen-and-Paper Proofs to Industrial Tools. Computing and Software Science, 2019
22. IEEE: IEEE Std 1012 - IEEE Standard for System, Software, and Hardware V&V. 2016

23. Kammueller, F.: Formal Modeling and Analysis of Data Protection for GDPR Compliance of IoT Healthcare Systems. SMC 2018
24. Khalastchi E., Kalech, M.: On Fault Detection and Diagnosis in Robotic Systems. ACM Computing Surveys 51(1): 9, 2018
25. Kuhn, T., et al.: A Simulator Coupling Architecture for the Creation of Digital Twins. ECSA 2020
26. Kramer, A., Legeard, B.: Model-Based Testing Essentials. Wiley, 2016
27. Laskey, M., et al.: DART: Noise Injection for Robust Imitation Learning. CoRL 2017
28. Luckcuck, M., et al.: Formal Specification and Verification of Autonomous Robotic Systems: A Survey. ACM Computing Surveys 52(5): 100, 2019
29. Nair, S., et al.: An Extended Systematic Literature Review on Provision of Evidence for Safety Certification. Information and Software Technology 56(7): 689-717, 2014
30. Natella, R., et al.: Assessing Dependability with Software Fault Injection: A Survey. ACM Computing Surveys 48(3): 44, 2016
31. Oxford UK Dictionary: method (online) https://www.lexico.com/definition/method, 2021
32. Pan, L., et al.: Cyber security attacks to modern vehicular systems. Journal of Information Security and Applications 36: 30-100, 2017
33. Pandit, H., et al.: Test-Driven Approach Towards GDPR Compliance. SEMANTiCS 2019
34. Parra, E., et al.: Advances in Artefact Quality Analysis for Safety-Critical Systems. ISSRE 2019
35. Paz, A., El Boussaidi, G.: A Requirements Modelling Language to Facilitate Avionics Software Verification and Certification. RET 2019
36. Rival, X., Yi, K.: Introduction to Static Analysis. An Abstract Interpretation Perspective. MIT Press, 2020
37. Sangchoolie, B., et al.: A study of the interplay between safety and security using model-implemented fault injection. EDCC 2018
38. Savary, A., et al.: Model-Based Robustness Testing in Event-B Using Mutation. SEFM 2015
39. Skoglund, M. et al.: Black-Box Testing for Security-Informed Safety of Automated Driving Systems. VTC 2021-Spring
40. Timperley, C.S., et al.: Crashing simulated planes is cheap: Can simulation detect robotics bugs early? ICST 2018
41. Tsachouridis, V.A., et al.: Formal analysis of the Schulz matrix inversion algorithm: A paradigm towards computer aided verification of general matrix flow solvers. Numerical Algebra, Control & Optimization 10(2): 177-206, 2020
42. US DoD: Defense Modeling & Simulation Coordination Office, V&V Technique Taxonomy (online) https://vva.msco.mil/default.htm?Ref_Docs/VVTechniques/, 2001
43. [Removed for double-blind review]
44. Yang, Y., et al.: Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid SCADA systems. SUPERGEN 2012