Trustworthy Autonomous System Development

JOSEPH SIFAKIS, Verimag Laboratory
DAVID HAREL, Weizmann Institute

Autonomous systems emerge from the need to progressively replace human operators by autonomous agents in a wide variety of application areas. We offer an analysis of the state of the art in developing autonomous systems, focusing on design and validation and showing that the multi-faceted challenges involved go well beyond the limits of weak AI. We argue that traditional model-based techniques are defeated by the complexity of the problem, while solutions based on end-to-end machine learning fail to provide the necessary trustworthiness. We advocate a hybrid design approach, which combines the two, adopting the best of each, and seeks tradeoffs between trustworthiness and performance. We claim that traditional risk analysis and mitigation techniques fail to scale and discuss the trend of moving away from correctness at design time and toward reliance on runtime assurance techniques. We argue that simulation and testing remain the only realistic approach for global validation and show how current methods can be adapted to autonomous systems. We conclude by discussing the factors that will play a decisive role in the acceptance of autonomous systems and by highlighting the urgent need for new theoretical foundations.

CCS Concepts: • **Software and its engineering;** • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → **Distributed computing methodologies**; **Artificial intelligence**;

Additional Key Words and Phrases: Autonomous system, trustworthy system development, dependability, machine learning, critical systems engineering, validation, simulation and testing, requirement and scenario specification

ACM Reference format:
Joseph Sifakis and David Harel. 2023. Trustworthy Autonomous System Development. *ACM Trans. Embedd. Comput. Syst.* 22, 3, Article 40 (April 2023), 24 pages.
https://doi.org/10.1145/3545178

1 INTRODUCTION

1.1 Characteristics of Autonomous Systems

Autonomous systems emerge from the need to automate existing organizations by progressive and incremental replacement of human operators by autonomous agents. They are very different from game-playing robots or intelligent personal assistants, are often critical, and should exhibit "broad intelligence" by handling knowledge to adapt to unpredictable and complex environments. In particular, this implies the ability to manage dynamically changing sets of possibly conflicting

Authors' addresses: J. Sifakis, Verimag Laboratory; email: Joseph.Sifakis@univ-grenoble-alpes.fr; D. Harel, Weizmann Institute; email: dharel@weizmann.ac.il.

1539-9087/2023/04-ART40 $15.00
https://doi.org/10.1145/3545178

goals. Furthermore, autonomous systems should be able to collaborate harmoniously with human agents to achieve common goals.

The development of trustworthy autonomous systems, as anticipated by the Industrial **Internet of Things (IoT)**, is considered a bold step toward closing the gap between human and artificial intelligence.

Autonomous vehicles are a topical case, emblematically illustrating the difficulties in moving from automation to autonomy. In contrast to the aerospace and rail industries, current trends in engineering autonomous vehicles have not followed a "safety by design" concept. To overcome the technical difficulties implied by model-based approaches, some industrial players have developed end-to-end **machine learning– (ML)** enabled solutions, for which there exist no rigorous validation techniques. Furthermore, in contrast to standard engineering practice, critical software can be modified by over-the-air updates.

Many believe that it is necessary to break with current development techniques, which constitute an obstacle to the acceptance of new technologies. Some show blunt realism, by claiming that we should charge ahead with new ideas, accepting the risks, as the benefits will be so great. Others reject rigorous approaches as inherently inadequate for such complex systems and show blind faith in ad hoc solutions. And then there are those who are wildly optimistic, believing that we already have the right tools and that full autonomy is only a matter of time.

1.2 Trustworthiness

The design of autonomous agents should not be limited to logical aspects. It must also address risk analysis, mitigation, and evaluation, focusing on the dangerous situations that can result from the interaction of the agent with its environment. We point out that existing techniques are not up to the task, because they assume an exhaustive analysis of the causes of the risk and the deployment of mitigation mechanisms at the design stage. This does not seem realistic due to the inherent complexity of the environment of an autonomous system. One trend in the quest to overcome these difficulties is to break away from the idea of correction at design time, and indeed new ideas are emerging that rely on runtime assurance techniques replacing static mitigation mechanisms.

The global validation of autonomous systems, regarded as ensembles of autonomous agents interacting to achieve collective goals, challenges rigorous validation techniques not only due to complexity issues but also by the lack of adequate modeling frameworks. Moreover, complexity arises not only from obvious metrics like the number of lines of code or the number of components but also from the temporal and spatial dynamism of the agents' interactions with the cyber-physical and human environments.

Simulation and testing remain the only feasible approach to assess the trustworthiness of an overall system. However, existing theory and techniques for testing hardware and software systems are model-based. They pursue well-defined objectives in the form of coverage criteria or achievement of test purposes that can be formulated and understood in a rigorous manner. Their application to autonomous systems would require the development of far more powerful modeling and testing techniques, as well as the formalization of adequate success criteria.

1.3 Can We Build Reliable Autonomous Systems?

Systems engineering is reaching a turning point, moving from small, automated, centralized, non-scalable systems with predictable environments to large, autonomous, distributed, reconfigurable systems with unpredictable, dynamically changing environments. The two main problems, which are of comparable importance, are as follows: (1) the design of autonomous agents that are able to pursue predefined goals in unpredictable environments and (2) the global validation of autonomous systems composed of an arbitrary (and dynamically changing) number of interacting agents.

Currently there are two different technical avenues for developing such systems, but both fall short of addressing the full autonomy challenge. On the one hand, traditional model-based critical systems engineering—successfully applied in, e.g., the automotive and aerospace industry—proves to be inadequate for autonomous systems, being unable to deal with the overwhelming complexity of the problem. On the other hand, end-to-end solutions based solely on ML—developed by large technology companies such as Waymo and Nvidia's autonomous driving platforms—exhibit a lack of explainability, which is a key barrier to their use for critical autonomous systems.

We provide a technical characterization of autonomous agents as the composition of characteristic functions, allowing, among other things, a clear distinction between automation and autonomy. This characterization shows that there is a big gap between automated systems and fully autonomous ones, which cannot be bridged by mere incremental improvement of existing solutions. For example, the inevitable integration into autonomous systems of modules that employ artificial intelligence makes current non-AI systems engineering techniques and standards vastly inadequate. Moreover, we argue that the vision of autonomous systems also raises difficult systems engineering issues that are not directly related to achieving intelligence.

One of the promising avenues to explore further is "hybrid design," which attempts to use the best parts of the two approaches, ML based and model based, and to find tradeoffs between trustworthiness and performance. A crucial issue is the need for a coherent integration of heterogeneous data and model-based components into a rigorous design flow.

1.4 Structure of the Article

The article is a continuation of our work, as described in previous publications with Assaf Marron [1, 2], advocating the vision of an Autonomics foundation for future generation autonomous systems. In line with this vision, the present article provides a more detailed analysis of the problems to be addressed and discusses possible technical paths toward solutions.

Section 2 explains why it is difficult to build autonomous systems by describing in some detail the many facets of the challenge.

Section 3 discusses issues related to hybrid design, assuming that the decision-making process is model based. It advocates the need for a hierarchical semantic model of the agent's environment, integrating concrete and symbolic data at different abstraction levels. The section closes with a presentation of non-trivial issues regarding the application of existing risk analysis techniques to autonomous vehicles and indicates possible avenues toward their solution.

Section 4 presents ideas for evaluating overall system properties using simulation and testing. We show how current methods and practices can be extended and adapted to autonomous systems and identify emerging technical requirements. We conclude with a discussion of criteria for corroborating conclusive evidence of trustworthiness.

Section 5 discusses factors that we feel will play a decisive role in shaping the future, such as the division of labor between humans and autonomous systems, the role and effect of regulations, and ethical issues. We then conclude by articulating the urgent need for new theoretical foundations, bridging the gap between machine and human intelligence.

2 REALIZING THE MAGNITUDE OF THE UNDERTAKING

2.1 Autonomy and Autonomic Complexity

The concept of autonomous systems has been around for more than two decades. It has motivated a large amount of research and development, which, for the most part, falls under the headings of autonomic computing [3], adaptive systems and autonomous agents. This work initially concerned purely software systems, but the concept of autonomous system studied here emerges from the

needs of the industrial IoT, focusing on systems of intelligent reactive agents that replace humans in complex organizations and facilities, such as realizing a cyber-physical system with human-level intelligence [4].

Replacing humans with machines suggests that the relevant test of intelligence cannot be just a textual question-and-answer imitation game, as in the Turing test. The kind of intelligence required here might be termed a *replacement game*, where the fact that a human has been replaced by an intelligent agent in a large scale multi-agent system or organization, complete with its entire rich environment, will be undetectable by a human tester.

2.1.1 Functional Characterization of an Autonomous Agent. An autonomous system consists of components of predefined types, *agents* and *objects,* sharing a common environment and coordinated so that their collective behavior meets given global objectives. The objects themselves are usually dynamic physical systems, whose states can change either through the actions of agents or internally.

An agent is a system (actually, a subsystem) that has the ability to monitor objects in its external environment and act based on their states, either alone or in coordination with other agents. The agent pursues a mission characterized by a set of specific goals that can change dynamically, depending on the state of its environment.

The environment provides an infrastructure and mechanisms implementing coordination rules that govern the interaction between the components, i.e., the agents and objects. In particular, these rules determine the connectivity between agents, as well as the observability and controllability of objects.

We propose an architecture for an autonomous agent, consisting of five key functions that work together to achieve autonomy. As we shall see, the architecture gives rise to a definition of autonomy as the capability of an agent to achieve a set of coordinated goals without human intervention, adapting to changes in the environment.

The agent is a reactive system [5] that receives and processes sensory information from its environment and computes commands, whereby actuators carry out actions that change the state of the environment.

Figure 1 depicts an agent for an autonomous vehicle. Its internal environment is the vehicle whose direction and speed are controlled by the agent. In its external environment, we see three vehicles, one pedestrian, and one traffic light. The agent processes the information concerning the environment, both internal and external, and issues commands to carry out the actions needed to achieve specific goals. The actions must take place within deadlines defined by the dynamics of the environment to ensure that the goals are achieved in a timely manner.

The agent combines five key functions, two of which are aimed at understanding environment situations (perception and reflection) and two carry out the decision-making (goal management and planning). The fifth function embodies the ability of knowledge management.

The agent is also equipped with a *knowledge repository*, where it stores acquired knowledge that is useful for identifying and managing sensory information, in particular. Knowledge includes concepts concerning objects in the environment and their properties, as well as methods for decision-making. In the example at hand, the concepts of "car," "pedestrian," and "lights" are required to "understand" the external environment. The repository for each of these concepts may contain information concerning their characteristic properties, so that the system achieves better predictability; e.g., it may know the maximum speed and acceleration of the specific type of car.

The *function of perception* receives sensory information from the environment (images, signals) and analyses it by distinguishing between concepts and possibly also relationships linking concepts that are stored in the repository. In our example, the sensory information from the external

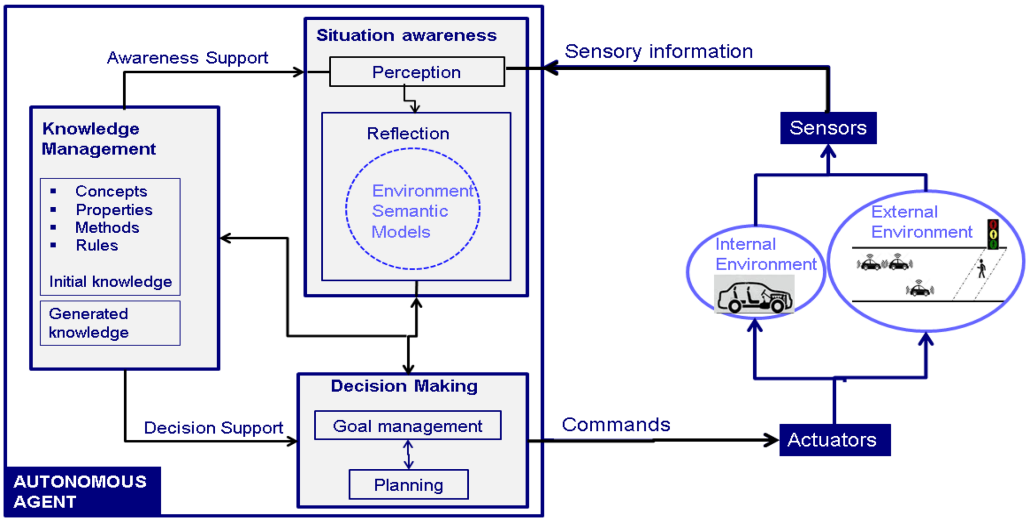

Fig. 1. Architecture of an autonomous agent for an autonomous vehicle, with its five key functions.

environment contains the three cars, the pedestrian, and the traffic light, with corresponding information on their positions and kinetic states. Regarding the internal environment, the sensory information concerns the kinetic state of the vehicle, such as speed and acceleration. Perception is usually accomplished using neural networks, which are currently the best technology for this purpose.

The perceived information is transferred to the *reflection function*, which is tasked with building a model of the external and internal environment of the system. This model has variables that represent states of the environment, such as the kinetic attributes of the obstacles and the state of the vehicle. The actions of the model are state changes that the controlled vehicle or the obstacles around it can perform. The model must be updated in real time to reflect the dynamic state of the environment as accurately as possible.

The *decision-making* module integrates two functions, using the environment model. The first of these performs *goal management* by selecting from a set of pre-defined goals a subset of compatible goals in relation to the current state of the environment model. This is really the system's strategy. The system's goals include both negative and positive ones. Negative goals concern the need to avoid undesirable states, such as safety goals regarding collision avoidance. Positive goals concern the need to achieve desirable conditions, such as optimizing passenger comfort, fuel consumption and arriving at a destination location.

We can also distinguish among various kinds of short-term goals, such as safety goals, medium-term goals, e.g., for maneuvering the vehicle to overtake other vehicles or drive through intersections, and long-term goals, such as completing an itinerary. Timely goal selection is crucial to system autonomy, because it is highly complex and requires computation times that must satisfy real-time response requirements.

Goal management is complemented by the *planning function*, which, after selecting the goals, determines the system's tactics. For each set of selected goals, this function calculates a sequence of commands to the actuators, which carry out corresponding actions to realize the commands. Thus, with regard to the collision avoidance goal, it must control speed and direction by adequately combining braking, acceleration, and steering wheel angle. For every kind of maneuver, the system must have at its disposal the appropriate tactics to achieve the corresponding goals.

Finally, the fifth key function is *knowledge management*, which manages and updates the knowledge on the repository. Knowledge is updated through the creation of new knowledge, concerning (1) the environment, e.g., new concepts based on accumulated knowledge from the analysis of model data, and (2) new goals for adapting to changes in the environment and changes in parameter values that are relevant to the choice between goals.

The agent's operation is cyclically repetitive. The cycle begins with perception, followed by reflection that updates the environment model. Then, decision-making takes place, possibly with the selection of new goals, and following up on the execution of tactics that were not completed in the previous cycle. These "leftovers" from the previous cycle can occur because the cycle's duration must be short enough to achieve some short-term goals (for safe driving this might be on the order of a tenth of a second), while achieving long-term goals may take hundreds of thousands of cycles (e.g., to reach a destination). Obviously, when new goals are selected in a cycle, they must be compatible with those that have already been selected, but which have not yet been achieved.

Thus, autonomous behavior of an agent, defined earlier as the ability to achieve its goals without human intervention, results from the combination of five mutually independent functions: perception, reflection, goal management, planning, and knowledge management. This characterization allows us to distinguish between automated and autonomous systems. Automated systems such as thermostats or lifts do not need any of these functions. They receive digital data from their environment, and each pursue very simple goals, achievable through static control policies.

2.1.2 Autonomic Complexity Issues. How difficult is it to build an autonomous agent? The previous functional characterization of autonomy guides us into distinguishing the following aspects of autonomic complexity.

(1) The complexity of perception characterizes the difficulty of interpreting the stimuli provided by the environment and of generating in due time the corresponding inputs for the agent's environment model. It has various sources, such as the ambiguity of the stimuli (admitting different interpretations) or their imprecision (fuzzy or noisy stimuli). Furthermore, this type of complexity is compounded by the volume of stimulus data needed to extract the relevant input information.
(2) The complexity of the reflection characterizes the difficulty of building a faithful and predictive semantic model of the agent's environment. This difficulty increases with two factors. The first is the lack of observability/controllability, which implies but partial knowledge of the agent's environment, and consequently limitations on building a faithful model. The second is the uncertainty about the agent's environment, which limits predictability. There are multiple sources of uncertainty, including time-varying load, dynamic changes due to mobility, and "bursty" events, as well as critical events like failures and cyber-attacks. Clearly, reduced observability is a source of uncertainty. Nevertheless, the uncertainty cannot be completely resolved by simply improving observability.

As explained above, the use of prerecorded knowledge can significantly improve situational awareness. For example, in self-driving cars, the reflection function can combine perception information with semantically rich prerecorded maps that show details of the road infrastructure to build an even more accurate semantic model of the environment.

(3) The complexity of decision-making concerns goal management and planning. For goal management, it reflects the cost of computing maximal sets of compatible goals for a given state. This computation may involve both qualitative criteria, such as priorities, and quantitative criteria, such as optimization of physical quantities. For planning, the complexity

depends on the type of goals and the complexity of the agent's environment model. As explained, goals may be as simple as non-violation of a constraint and more complicated such as reachability of a condition or performance over a given time period.

(4) The complexity of knowledge management has two aspects. One arises from supporting situational awareness and decision-making by providing appropriate knowledge, for example, for a given type of obstacle identified, to provide the reflection function with information from the repository about its relevant properties and enrich the semantic model or to compute estimates of parameters used in the decision-making process, such as delays, traffic density, and risk parameters.

The other aspect consists of discovering new knowledge needed to face entirely new situations, not foreseen at design time, e.g., situational awareness can be improved by the discovery of new concepts or decision-making can be enriched by new objectives adapting to a changing environment.

2.2 Systems Engineering Complexity

As already explained, the construction of autonomous agents involves difficult system engineering problems. We can define a concept of *system complexity* that characterizes the difficulty of building a system from components, autonomous or not [6]. System complexity is the product of two factors: (1) *reactive complexity* [7], which characterizes the difficulty of building the components constituting the system, and (2) *architectural complexity* [6], which characterizes the difficulty of achieving the desired coordination.

We show that component and architecture characteristics place autonomous systems among the most difficult to build from a systems engineering perspective.

2.2.1 Reactive Complexity. Reactive complexity characterizes the complexity of the interaction between a component, e.g., an agent or an object, and its environment. It is independent of the space or time complexity that measure the amount of computational resources required for its operation.

We propose a classification of components based on their reactive complexity.

The simplest kind of components are *transformational components*, where the relationship between input and output values is sufficient to characterize their behavior. The computation is performed in batch mode, without reference to any operating environment. Such components are often software systems ignoring real-time constraints, with simple and well-defined environments.

Streaming components compute functions on data streams, like encoders or signal processing systems. For a given input value stream, they compute a corresponding output stream. Their goals concern mainly functional correctness, with specific time-dependent properties such as latency.

Embedded components continuously interact with a physical environment to ensure global properties. These are typically mixed HW/SW systems, where real-time behavior and dynamic properties are essential for correctness.

Cyber-physical components have the highest reactive complexity, as they seek a tight integration between computers and their physical environment [8]. They are embedded components integrating in their internal environment objects that are exclusively under their control. The description of their behavior requires both discrete and continuous variables representing the states of the integrated objects.

Autonomous systems, such as self-driving cars, smart grids, and smart factories, should be built from cyber-physical components to meet the compelling needs for mobility and tight integration with electromechanical devices.

2.2.2 Architectural Complexity. Architectural complexity reflects the difficulty of modeling, analyzing, and implementing coordination mechanisms involved in the architecture of a system. Following a classification proposed in Reference [6], we list below some representative cases of increasing complexity.

Static architectures involve a fixed number of components, agents or objects, with fixed positions, e.g., a smart building system architecture with fixed microcontrollers and connections to electromechanical equipment.

Parametric architectures have an arbitrary, initially known number of "plug-in" types of components for fixed coordination patterns, e.g., token ring architecture or cellular architecture.

Dynamic architectures are parametric architectures with dynamic creation/deletion of instances of component types, such as client-server architectures. It should be noted that modeling and reasoning about the properties of dynamic architectures requires languages that allow parametric and generic description, such as higher order logics.

Mobile architectures are dynamic architectures where, in addition to temporal dynamism, there is spatial dynamism: the coordinates of components can change dynamically, for example in mobile telecommunication systems.

Self-organizing architectures are mobile architectures where the coordination rules of the components depend on their position in a structure. For example, for self-driving cars and swarm robots, the coordination of agents changes with time and space but also depends on their position in an organization, e.g., platooning architecture or stigmergy architecture.

2.2.3 Some Conclusions. The proposed classification clearly shows the systems engineering issues underlying the development of autonomous systems. These arise from the combination of physicality and computation and include component heterogeneity and composability and multi-scale and multi-dimensional modeling and analysis.

Self-organization is difficult to model and analyze, because it involves three different types of dynamism: temporal, spatial, and organizational. We already know that verification techniques successfully applied to the verification of static systems suffer from serious undecidability limitations when we move to parametric architectures, even if they have finite state components, e.g., Reference [9]. Thus, formal verification of the global properties of autonomous systems is not feasible.

Note that dynamism and distribution raise additional concerns. Autonomous systems involving dozens of agents spanning a large area will not be free of undesirable emergent properties, even if each agent considered separately is shown to be safe. Consider the trivial deadlock caused by four uncoordinated vehicles waiting for each other at a four-way stop, strictly adhering to the right-of-way rule. Determining and preventing the emergence of hazardous situations at design time is a difficult problem.

3 TRUSTWORTHY AUTONOMOUS AGENT DESIGN

Trustworthiness is a transversal design issue. It is not limited to purely functional correctness. A system is deemed trustworthy if it behaves as expected despite design errors, hardware failures, and any kind of potentially harmful interaction with its human and physical environment, including misuse, attacks, disturbances, and any kind of unpredictable events [10].

In this section, we explain the limitations of current model-based approaches and explore the possibility of using them in the agent design flow for the development of those functions for which their application, if possible, would be highly beneficial. In particular, we discuss two underlying problems with model-based decision-making and identify avenues for achieving trustworthiness in the face of hazards and incidents.

3.1 Current Limitations of Model-based Approaches

Model-based systems engineering has been successfully applied to develop systems with guaranteed trustworthiness, such as automotive, aerospace, and production systems. There are many reasons that existing methodologies are not applicable to the development of autonomous systems. Rigorous methodologies, recommended by standards such as ISO 26262 for functional safety of road vehicles [11], are based on the V-model [12], a prescriptive framework that assumes a top-down system design and bottom-up validation flow.

A strong assumption underlying these methodologies is that the system requirements are known from the start and can be clearly formulated and understood. Furthermore, the requirements must be decomposed into properties satisfied by the system components in a top-down refinement process.

This assumption does not seem realistic for our purposes, even for non-autonomous complex systems, since such systems are not designed from scratch; they are often built by incrementally modifying existing systems and largely reusing components. In fact, projecting global system requirements into the components of a system architecture is a non-trivial problem. Furthermore, the V-model assumes a possibility of compositional validation after the completion of the implementation and that the correctness of the system can be established by gradually moving from the components to the overall system validation.

Post-design verification is not realistic for complex systems, where it is important to detect design errors as early as possible. For these reasons, modern software engineering has moved away from the V-model to so-called agile methodologies, which consider that coding and design should go hand in hand: Designs should be modified to reflect adjustments to requirements [13]. However, the problem of finding rigorous methodologies that escape the V-model straitjacket and meet current needs remains entirely open.

For autonomous systems, not only is the V-model not applicable, but the use of non-explainable ML components prevents the full application of model-based methodologies. These limitations are particularly reflected in the current lack of standards for autonomous vehicles.

As explained earlier, a compromise would be to adopt a hybrid approach that integrates model-based decision-making modules with data-based components. The construction of such modules could build on the well-established results of model-based adaptation, which would also provide high confidence guarantees.

3.2 Model-based Decision-making

We discuss two key issues for model-based decision-making. The first is hierarchical decision-making, and the second is the representation of the external world using maps.

3.2.1 Hierarchical Control Architecture. The decision-making of an autonomous agent involves managing at least three different types of goals.

Short-term goals, which are subject to strict real-time and safety constraints. They require the system to stay away from dangerous states. For example, for an autonomous vehicle, they would aim to avoid collisions by keeping their distance from obstacles within certain limits or by following a pre-defined trajectory. For smart grids, these goals mostly concern robustness, i.e., the ability to provide stable and continuous energy flows.

Medium-term goals, which concern the transition between predefined operating modes, to adapt to dynamically changing situations. For an autonomous vehicle, these goals require performing maneuvers such as overtaking or crossing intersections of various types. For a smart grid, they are about supporting the integration of renewable electricity and the system's ability to

reconfigure itself to adapt to changing demand. Note that medium-term goals imply dynamic system reconfiguration and adaptability under specific time constraints.

Long-term goals, which are designed to satisfy various types of non-critical properties, including optimizing criteria or meeting given target conditions. For an autonomous vehicle, this might be completing a trip by reaching a destination or optimizing fuel consumption. For a smart grid, it may be asset optimization, cost reduction, and operational efficiency.

It should be noted that medium-term goals are the most difficult to achieve, because they are subject to safety and system controllability requirements under the uncertainty that results from unpredictable environments. In contrast, short-term goals usually lend themselves to formalization and can be achieved by applying, e.g., well-established control theory and technology. This is typically the case for trajectory tracking, collision avoidance, or network robustness. However, long-term, non-time-critical goals have less stringent requirements and can be met using a variety of strategies.

This distinction between goals leads naturally to a hierarchical control paradigm for autonomous agents. The lowest level is responsible for achieving short-term goals, the second involves tasks aimed at achieving medium-term goals, and the third level deals with achieving long-term mission goals. Clearly, the three levels operate on increasing time scales as one moves up the hierarchy.

The adoption of such a hierarchical control paradigm is advocated by the seminal work at NIST [14, 15], which proposed the 4D/**Real-time Control System (RCS)** reference architecture described in a series of methodological and technical papers. Nevertheless, to our knowledge, it is not clear that 4D/RCS has ever been fully applied beyond modest demonstrations. Recent publications, e.g., References [16–19], address the analysis of such architectures, but we still lack conclusive results as to the implications and risks of applying the principle to real-world autonomous systems.

The main difficulty in implementing the hierarchical control principle lies is the timely and harmonious coordination of the three levels, since control involves a complex top-down and bottom-up flow realizing the interaction of processes with different underlying dynamics. The top-down flow ensures consistency and controllability while the bottom-up flow ensures observability. In addition, the dynamic change of a goal at one level must be consistent with the goals pursued at other levels. For example, if the execution of a maneuver requires acceleration, then this must be consistent with keeping the vehicle on its intended trajectory.

Note that an underlying assumption of this approach is that agent autonomy is realizable as a dynamic composition of a set of basic tasks, each allowing the achievement of a specific goal. It is analogous to, and consistent with, the idea that human autonomy results from the intelligent combination of skills. For example, the ability to drive is the combination of skills involving keeping a safe distance from obstacles, maintaining a certain trajectory, performing various maneuvers, and so on.

An advantage of the hierarchical control paradigm is that it is possible to analyze and verify separately that each subsystem allows achieving the corresponding goal. For example, we can prove that the collision avoidance system or the overtaking protocol are correct under certain integration conditions applied at the agent architecture level. Of course, it remains to be demonstrated that the desired agent behavior can be achieved by properly integrating a set of basic goal-achieving tasks into the hierarchical architecture.

Often the integration conditions we want to assume are ones relevant to the AI/ML parts of the system we are developing. Given that one cannot verify these parts (in the formal mathematical sense), we could advocate "relative verification," which means that we verify the non-AI

components of the system (or portion thereof) that we want to verify, while assuming the correctness of the AI parts. This yields the "relative correctness" of the system. Of course, this necessitates somehow defining what we even mean by an AI/ML component being correct, which would presumably be a probabilistic statement about the outcomes. However, from a purist's stand, this is the best kind of verification one can strive for in the context of an autonomous system with components of both kinds.

3.2.2 Map-based Representation of the Environment. Autonomous systems are overwhelmingly distributed systems, too, with agents deployed in a variety of spatial locations within some physical environment. As explained, the behavior of an agent depends strongly on the physical context in which it evolves. Decisions are made based on a semantic model of the environment, which takes into account the relevant geometric features and the presence of objects or agents therein.

Thus, an agent's semantic model is an abstraction of the physical environment as perceived by the sensing devices. It can be represented by a suitable map, enriched with observable state attributes of objects or agents in the agent's neighborhood.

Nevertheless, relying exclusively on models built from local sensory information is often not sufficient for effective autonomy. To manage medium and long-term goals and anticipate external changes, a broader view of the environment is needed. For example, autonomous cars can use maps stored in a repository enriched with online traffic information, which would be similar to, but vastly richer than, the maps used in current car navigation systems. These maps are essential to indicate long-term goals such as the route taken by the vehicle or to anticipate situations beyond the agent's visibility. For example, knowing the geometric characteristics of a nearby roundabout allows for better preparation of the corresponding maneuver.

Consistent with the distinction between the three types of goals, the external environment maps we need should themselves be designed at three different levels of abstraction. High-level maps are used to represent long-term goals, such as the entire mission goals, and provide a description of the area where the autonomous agent can operate, e.g., for autonomous vehicles, a map of the road network where routes can be indicated.

The middle-level maps describe the environment at a scale that provides the details necessary to implement the medium-term goals. These involve features of the environment that require a specific agent control policy. For autonomous vehicle maneuvers, we need maps providing detailed geometric descriptions of roads with their lanes and of intersections with their entrances and exits and associated traffic rules.

Finally, the low-level maps should provide a detailed description of the current external environment with all relevant information for efficient decision-making. This description can be obtained by merging sensory information and existing detailed maps and must be updated in real time.

The coherent integration of the three modeling levels raises some non-trivial issues related to the connection between the concrete map representations obtained by the fusion of sensory information and the available symbolic and semantically rich pre-existing map representations.

For effective situational awareness, the perception function must be able to distinguish not only simple objects, but also relevant object models, and for this, the use of pre-existing map models may be required, too. For example, the autopilot of an autonomous vehicle must be able to unambiguously identify different types of intersections and their associated signaling equipment.

There have been extensive efforts to define adequate semantic map models for autonomous vehicles. These include proposals for the standardization of semantically rich map models [20, 21]. Other work focuses on formalizing maps using ontologies and logics and applying reasoning mechanisms to check the consistency of descriptions and their accuracy with respect to desired properties [22–25].

In summary, building faithful semantic models for autonomous systems is a challenging problem arising from the fact that autonomous systems will be deployed in physical environments and their interactions will take place there. This requires a multi-scale representation of the environment in the form of maps reflecting the scope and granularity of different objectives. Moreover, it poses the problem of linking in real time the concrete knowledge generated by machine learning techniques on the perceived agent's environment with pre-existing and semantically rich symbolic knowledge.

3.3 Design for Dependability

Critical system development involves design for dependability, which aims to ensure that if certain assumptions about the system's nominal behavior are violated for any reason, then the system will be resilient and can avoid or circumvent hazardous situations. Dependability is not a black or white concept, in contrast to correctness with respect to a set of well-defined specifications. It takes into account failure rates of physical components, rare events, system misuse, and malevolent actions and is usually characterized by a set of probabilistic attributes such as reliability, availability, maintainability, and so on [26].

Design for dependability involves a three-step flow: risk analysis, risk mitigation, and risk assessment.

Risk analysis aims at providing answers to the following questions: What are the consequences of system hazards, what can go wrong, and how likely are these to occur [27]. Briefly, there is a large number of risk analysis techniques available to study the links between the different causes of risk and their potential effects. They cover a wide range of methods and practices, from the simplest, such as fault tree analysis [28], to the most sophisticated, involving architecture analysis, such as STPA [29] or dataflow analysis, such as FPTC [30]. The causes of risk can be "internal," such as design errors or software bugs, or external, due to human misbehavior (malicious or not) or natural causes, like failures or disasters. Their effects can be hazards compromising the safety, the security, or the performance of the system.

Risk analysis cannot be fully automated, and it requires common-sense engineering skills and a thorough understanding of the system's behavior and its interaction with the environment. It requires good systems engineering expertise, and obviously it can be tedious, as it requires thorough case-by-case analysis.

Risk analysis techniques, successfully applied to automated systems operating in aircraft or factories, are difficult to apply to autonomous systems due to the complexity and unpredictability of that latter's physical and human environment. For example, a failure typology for light vehicles published in a DOT document [31], lists 37 cases, many of which are difficult to analyze, because they are the results of imponderable events or human and animal actions

Existing risk mitigation methods suffer from similar limitations. They involve the systematic design of mechanisms to address the hazards/threats identified in the risk analysis by implementing the so-called detection-isolation-recovery mechanisms designed to ensure resilience [32]. For each type of hazard, a detection mechanism is implemented with associated isolation techniques to contain its effect until a mitigation mechanism can take over.

Isolation techniques are of a variety of types, ranging from partitioned architectures, so that the memory and processing time of one partition is not affected by another faulty partition, to firewalls, cryptography, and privileged access management. Similarly, recovery techniques range from the use of massive redundancy in hardware architectures, such as TMR, to rollback or roll-forward for software systems to reconfiguration techniques.

As already explained, these techniques are enumerative in nature and imply high combinatorial complexity for unpredictable environments. In general, their application is prohibitive for

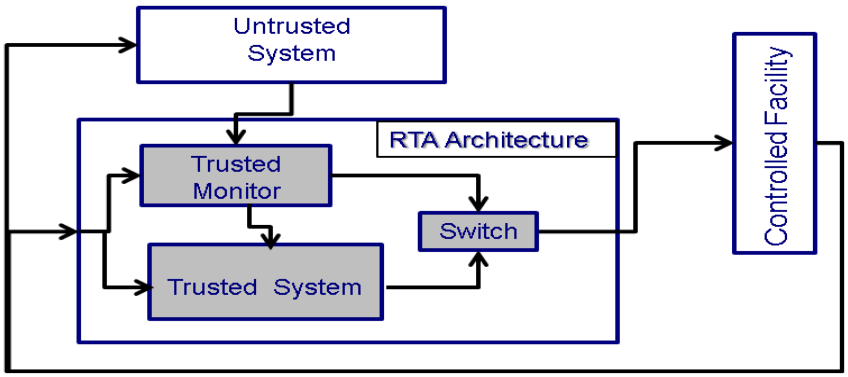

Fig. 2. Runtime assurance architecture.

autonomous systems. It can be considered for risks that are easy to identify and characterize, such as those caused by failures in electromechanical systems, but these represent only a small fragment of the risk factors. For example, Reference [33] identifies nine categories of risk for autonomous vehicles, including 176 different types of hazards.

For hazards that are difficult to predict and assess, the tendency is not to try to detect and mitigate risks through dedicated mechanisms statically at design time but rather to achieve resilience using assurance mechanisms at runtime.

Figure 2 illustrates a materialization of this idea with a runtime assurance architecture [34]. This architecture involves a trusted Monitor that detects deviations from the nominal behavior of an untrusted system, specified by a set of essential system properties. The Untrusted System can be of any type, such as an autonomous agent or a neural network, which controls some Facility.

The Trusted Monitor receives the output of the Untrusted System and is able to detect online any violation of the essential system properties. Upon detection of the offending event, it triggers a switch that replaces the output of the Untrusted System with the output of a Trusted System that can take over and provide some minimal service, so as to keep the Facility safe. When the Trusted Monitor diagnoses the recovery of the Untrusted System, it prompts the switch to swap outputs back and resort back to normal behavior.

Note that the runtime assurance architecture extends the Simplex paradigm used in fault-tolerant systems. The Simplex architecture differs from this in that when a hazard is detected, the system is directed to a fail-safe state and shuts down [35].

Runtime assurance is also consistent with the idea of hybrid control for collision avoidance, which has been proposed for self-driving cars, where an unreliable optimized controller is monitored by a provably safe controller [36, 37].

The application of the runtime assurance paradigm involves considerable technical difficulties that should not be underestimated, and it requires (like several other ideas in this article) non-trivial further research. Although the Untrusted System is treated as a black box, the development of trusted components must be model-based and then properly validated. Note that if the essential properties are formalized in linear temporal logic, then it is possible to use well-established online verification techniques to automatically generate the Monitor [38, 39].

It is important to realize that detection of property violation must occur early enough so that the Trusted System can still mitigate the hazard. This requires a deeper understanding and analysis of the Facility's dynamics to anticipate the hazard and adequately counter it by mitigating its effect online.

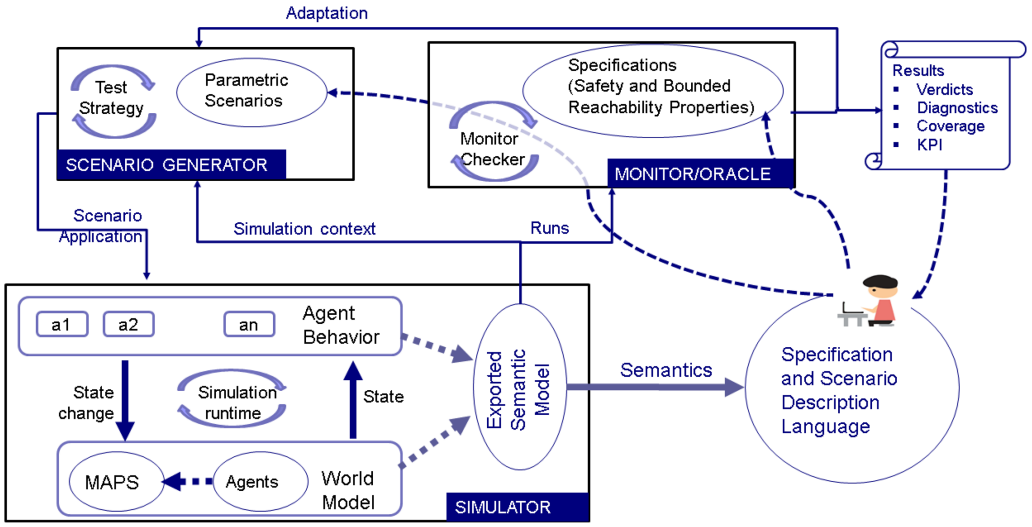

Fig. 3. Simulation and test architecture for validating autonomous systems.

4 AUTONOMOUS SYSTEMS VALIDATION

There is a significant gap between the state of the art in system validation and verification and the validation needs of truly autonomous systems. How can existing results be integrated and extended into a rigorous validation methodology aimed at providing conclusive trustworthiness evidence? We provide elements of an answer by analyzing various aspects of the validation problem and related methodological and technical issues.

Simulation and testing seem to us be the only feasible way for validation, especially when we consider an entire autonomous system. And this is regardless of the approach used for agent development—model-based or data-based. In the industrial world, there is an increasing awareness of the importance of simulation techniques for validation purposes, especially for autonomous vehicles.

For self-driving cars, a sufficiently low risk ratio per simulated mile is currently considered a near guarantee of safety [40]. However, this argument is not defensible, for the simple reason that not all simulated miles are equally effective. It is necessary to explain how simulated miles relate to "real miles." This requires a deeper understanding, through rigorous modeling, of the extent to which all relevant system configurations have been explored.

Among other things, we discuss what would be required of a solid and useful validation methodology, arguing that it must rely on model-based criteria by carefully and rigorously exploring a meaningful and relevant sample of system configurations.

4.1 Simulation and Testing Architecture

Figure 3 describes a general simulation and testing architecture for validating autonomous systems against given specifications, suitably formalized as safety properties or bounded accessibility properties. The architecture integrates three tools: (i) a Simulator, (ii) a Scenario Generator, and (iii) a Monitor.

The simulation is driven by actions generated by the Scenario Generator. Its runtime coordinates the execution of an autonomous system model and generates runs that are then verified online by the Monitor. The model should ideally consist of two entities: (1) a world model that includes a map

representing the global static environment, as well as the system agents with their state attributes, and (2) behavioral models of the agents and their possible interactions.

The model thus defines a dynamic system involving agents and objects that interact in a static environment represented by a map. The states of the agents and objects are defined by sets of attributes concerning their internal state and their kinetic state. However, unlike static systems whose global states are tuples of states of their components, the global states of autonomous systems are configurations involving not only the states of their components, agents, and objects but also contextual information depending on their position on the map [25]. This information is needed to determine the possible interactions between the components, as well as the interactions between the components and their physical environment.

For example, a configuration of an autonomous transport system contains the states of its vehicles and their positions on a map with its signaling equipment. Thus, the behavior of a vehicle depends on its state but also on the constraints induced by the possible interactions it may have with other vehicles and by the interpretation of the physical environment with the associated traffic equipment.

The simulator operates in a cyclic manner. At the beginning of a cycle step, it provides each agent with the current relevant information about the configuration of its neighborhood, as well as requests to execute control actions of a scenario. At the end of the step, it calculates the new configuration resulting from the state changes reported by the agents. We assume that the Simulator exports to the other tools runs of system configurations, as well as implementations of basic predicates, variables, and actions, via an interface, say, an adequately defined API.

A central and crucial part of this setup is the Scenario Generator, which applies a test strategy that drives the entire simulation process. The scenarios it generates are coordinated sequences of actions, to be executed by the agents in the Simulator. Thus, scenarios play the role of test cases intended to drive the simulated system toward specific configurations, for example to explore high-risk configurations, or to meet specific coverage criteria, as discussed below

The Monitor continuously receives the generated runs and applies runtime verification techniques to check online that they meet the given specifications. In addition, it can provide diagnostics and key performance indicators used by the Scenario Generator in its testing strategy. As explained, for any monitored run, the specifications express either safety, e.g., that all configurations of the run satisfy some safety condition on configurations, or bounded reachability, e.g., that there exists some configuration of the run that satisfies a desirable condition.

The architecture is an adaptive testing environment: using common terminology, the Simulator corresponds to a Unit under Test, the Monitor is an Oracle, and the Scenario Generator plays the role of a Test Case Generator.

4.2 Requirements for Simulation

We are of the unequivocal opinion that simulation is of paramount importance for validation of autonomous systems, and that it covers a wide variety of aspects—from the purely technical to the theoretical. We see three main requirements for a good simulation:

(1) Realism: The agent's behavior and environment must look real, involving suitable graphics that relay the situations and the dynamics thereof in a way that is true to reality. For modest examples of realistic-looking simulation models for biological systems see, e.g., References [41, 42].
(2) Semantic awareness: From the description of the simulated system model, it should be possible to extract a semantic model, including a well-defined notion of system configuration. This allows one to define predicates on system configurations that will be used

by the monitor to verify online system properties. Note that semantic awareness implies that the simulated models respect fundamental properties of time and space, because configurations have time and space attributes. For example, if there are two different runs leading from one configuration to another, then the travel time and distance along these sequences must be comparable.

(3) Multiscale/multigrain modeling: We have identified at least three different levels of modeling abstraction corresponding to the different types of goals pursued. To analyze and validate the behavior of the system with respect to the properties relevant to these goals, it is important to be able to adjust the granularity of the simulation appropriately. Coarse-grained simulation is sufficient for mission-related properties while fine-grained simulation is essential for the lower level dynamic properties of an agent, taking into account the details of the physical space.

To meet the above requirements and to cope with the complexity of the simulated systems, we need specific execution infrastructures such as HLA or FMI, integrating the execution of specific simulation engines at different space and time scales [8].

Note that realism and semantic awareness are not easy to reconcile. Industrial simulation environments often favor realism, because they are built on top of game platforms that have not been developed with semantic awareness in mind. Obviously, the configurations of the exported semantic model must take into account only essential aspects of the environment. They can ignore the details that are not involved in the formalization of the properties to validate, even if these may affect the system's dynamics and hence the results of the simulation. For example, weather conditions may have an impact on vehicle dynamics while they are not relevant for the expression and validation of basic traffic rules.

4.3 Scenario Generation and Testing

The Scenario Generator receives configuration runs exported by the Simulator and applies rules to control the evolution of the simulated system, e.g., to explore particular situations or to enforce given properties. In practice, scenarios give rise to sequences of control actions that modify parameters of the agents in the simulated system and hence their states. For instance, if the agents are vehicles, then the parameters can be their initial positions on the map, or their itineraries,

Obviously, the scenarios must be compatible with the behavior of the agents specified in the simulator and their operational context. For example, if a scenario requires a vehicle to accelerate, then this must be compatible with its collision avoidance policy. In other words, the Scenario Generator must apply sequences of actions gradually and carefully, following an adaptive policy that takes into account the configuration of the simulated system at each step.

There has been a large amount of work devoted to the study of scenarios for autonomous driving systems. This includes the Open-SCENARIO proposal [43] for the description of scenarios for driving and traffic simulators, as well as papers on the use of scenarios in testing and validation methods.

The work in Reference [44] proposes a visual formal specification language for capturing scenarios, termed live sequence charts, which was inspired by message sequence charts. Possible applications of this language for the specification and testing of autonomous vehicles were proposed in Reference [45]. See also the executable version of this language, in Reference [46]. The work in Reference [47] presents an approach for automated scenario-based testing of the safety of autonomous vehicles, using metric temporal logic. Finally, the probabilistic language Scenic [48] for the design and analysis of cyber physical systems uses scenarios to control and validate simulated systems of self-driving cars.

Is it possible to extend conventional testing techniques to autonomous systems? Testing strategies should aim to optimize criteria, such as coverage, or to test functional properties. While for software systems coverage can be defined as the ratio of source code exercised during the execution of test sequences, it is not clear how a similar criterion can be defined for autonomous systems. One of the main reasons for this is the non-availability of a comprehensive definition of the requirements and possible scenarios, from which to derive coverage criteria. Furthermore, functional testing assumes the existence of system behavioral models.

4.4 Monitoring and Validation

The Monitor applies online verification techniques to check whether the configuration runs generated by the Simulator satisfy the monitored properties. These characterize the evolution of agents on a map, which plays the role of a kind of "global resource" shared by the agents.

The validation of autonomous systems has motivated work on logics and associated verification techniques. Nevertheless, existing results on the validation of reconfigurable dynamic systems can be specialized to autonomous systems, for example [39, 49].

The work in Reference [50] proposes a formalization of traffic rules in linear temporal logic and applies runtime verification to check that the maneuvers of a high-level planner are consistent with the rules. In References [51, 52], a set of traffic rules for highway scenarios is formalized in Isabelle/HOL. It is shown that traffic rules can be used as requirements to be met by autonomous vehicles, and a verification procedure is proposed. In Reference [53], a formalization of traffic rules for uncontrolled intersections is proposed in first-order logic, and the rules are applied by a simulator to control traffic at intersections.

Consistent with prevailing approaches, the specification language we need can be a linear first-order temporal logic generated from a set of appropriately chosen atomic propositions. These are predicates expressing relations between the states of components, agents or objects, or relations between components and their static environment. For example, for autonomous vehicles, we need distance predicates, e.g., constraints on the distance between two vehicles, or location predicates, e.g., a vehicle should wait in front of a stop sign [25]. The API of the Simulator must provide implementations of the atomic propositions that can be evaluated on exported system configurations.

Note that quantification is necessary to express genericity and parameterization of variables on agent domains or static environment structures. In particular, it allows one to express properties that hold for all agents or for all environment models of a certain type, or, dually, for at least one thereof. For example, traffic rules for roundabouts involve a universal quantification on vehicles and on this type of junction.

Checking that a configuration run meets a given specification involves two steps. The first deals with the elimination of quantifiers in the formulas, taking account of the instances of the agents and the context. The second step deals with the generation from the resulting propositional temporal logic formula of a corresponding finite state automaton that characterizes the runs satisfying the formula and can be used for online verification of the observed behavior [38, 39, 49, 50].

To reduce the complexity of the space of configurations to be explored, we need structuring criteria inducing useful property-preserving abstractions. One solution to deal with this complexity is metamorphic testing [54], which can allow a drastic reduction of the number of test cases to be considered. It consists in defining "metamorphic relationships" on configurations and, by extension, on scenarios, too. Intuitively, if two scenarios are related, then they both satisfy the same set of properties. Thus, executions conducted by related scenarios from the same initial configurations should be indistinguishable by the Monitor. Thus, testing but one scenario per class can result in coverage guarantees for all its scenarios.

Another approach applied to reconfigurable systems [39, 49], consists of associating with a configuration a hypergraph representing the possible interactions between its components, objects, or agents. Given a configuration, the vertices of the corresponding hypergraph are the components involved in the configuration and the hyper-edges represent the possible interactions between the components. Intuitively, a set of components can interact if their behavior is constrained by the same property. It is therefore possible to group together configurations that have the same associated hypergraphs if the states of their components are "close" enough. For example, two configurations involving three vehicles traveling one after the other on a lane have the same hypergraph induced by the traffic rules, and they can thus be considered "equivalent" under certain conditions on their kinematic states. This simplification, used for the efficient monitoring and analysis of reconfigurable systems [55], can be advantageously applied to autonomous systems, too, as proposed in Reference [25].

5 THE WAY AHEAD

Success in achieving widespread acceptance and deployment of autonomous systems will depend on our ability to adequately address not only the multifaceted scientific and technical challenges but also the related societal and ethical issues, striking the right balance in the symbiosis between humans and machines.

5.1 Moving from Automated to Autonomous Systems

How can we increase the degree of autonomy in a design and what are the underlying technical challenges? The proposed functional characterization of autonomous agents provides some insights into this issue, which is at the heart of the transition from automation to full autonomy.

SAE International proposes a six-level hierarchy of driving automation, ranging from manual to fully autonomous systems [56]. The first three levels, 0 to 2, characterize degrees of automation, where a driver is ultimately responsible for driving the vehicle, possibly assisted by **advanced driver-assistance systems (ADAS)**. In levels 3 to 5, the autopilot is responsible for driving the system. Level 3 requires supervision of the autopilot by a human driver, level 4 allows autonomous driving in geo-fenced environments, while level 5 provides full autonomy without restriction.

This hierarchy suggests that the difficulty of moving from one level to another may be progressive. However, there is a major gap between automation and autonomy. At levels 0 to 2, ADAS systems operate under the control of the driver and can be turned on and off at any time, except in very specific critical situations. In contrast, at levels 3 to 5, the autopilot is responsible for driving, which is no small difference at all. ADAS cannot merely evolve progressively into autonomous driving systems, because automation and autonomy are two very different issues, as explained in Section 2.

In addition, the supervision of vehicles on autopilot prescribed by level 3 is proving to be a very risky idea. Safe collaboration between autonomous systems and humans raises symbiotic autonomy issues that go far beyond traditional HMI. When the autopilot proactively requests human intervention, the human must have the information and the time to understand the situation and act appropriately. Furthermore, if the supervisor realizes that something is wrong, then disengaging the autopilot or overriding the machine's decisions should result in situations controllable immediately by the human.

Finally, there is a significant gap between level 4 and level 5. Indeed, most of the complexity factors of autonomy stem from the lack of confidence in the perception function and the unpredictability of the external environment. Level 4 autonomy minimizes the risks associated with these factors. Driving in geo-fenced environments greatly simplifies the perception problem due to far greater predictability and vastly fewer possible interactions and configurations.

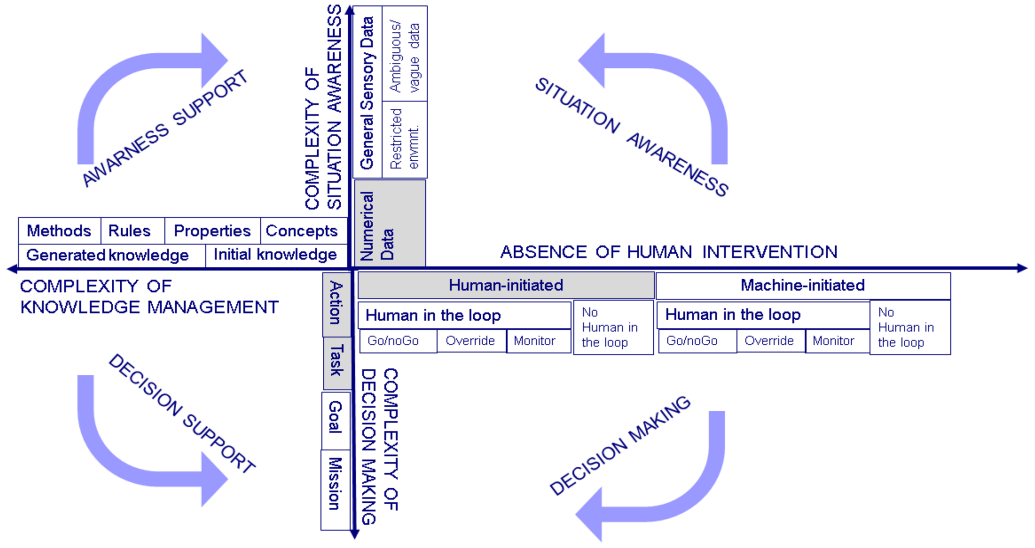

Fig. 4. The automation/autonomy space.

Furthermore, for such environments, extensive instrumentation can be used to improve the quality of perception. For example, platooning of trucks on highways seems to be feasible in the near future.

Figure 4, which is consistent with, and in fact reflects, our characterization of autonomy, shows how varying degrees of autonomy can be achieved through the interplay of four parameters: (1) complexity of situation awareness, (2) complexity of decision-making, (3) absence of human intervention in operating the system, and (4) complexity of knowledge management. The grey intervals delimit pure automation: the environmental stimuli are digital data and the operation of the system under human control, possibly assisted by automated systems.

The degree of autonomy increases as one moves from the automation zone away from the origin, as when, for example, one moves from digital sensory data to quality data, and then to general data that may be ambiguous and/or vague. Autonomy also increases as one moves from human to machine control, with the human remaining in the loop, with or without the possibility of intervention.

When the operation of the system is controlled by the machine, decision-making concerns management at the level of goals or even the entire mission. Finally, autonomy is reinforced by the ability to manage knowledge, as we move from fixed knowledge to the case where the system generates knowledge, creating in the process new concepts and objectives.

In conclusion, although there is a clear distinction between automation and autonomy, there are different ways to increase the autonomy of a system. We note that similar ideas have been presented in References [57, 58] to study the degrees of autonomy of assistants.

5.2 Matching Human Situation Awareness

Humans are far superior to machines in understanding complex situations, simply because they possess common-sense knowledge and reasoning [59]. The human mind understands situations by combining (1) bottom-up reasoning, from the sensor level to concepts, and (2) top-down reasoning, from concepts to perception. It is equipped with a semantic model of the world built progressively since our birth. Common-sense knowledge relies on this model to understand natural language and to deal with real-life situations.

However, machine learning only goes from the bottom upward. For example, it has been reported that an autopilot mistook the moon for a yellow traffic light [60]. This type of error could never happen to humans, simply because sensory information is contextualized using common-sense logic (traffic lights cannot be in the sky).

Similarly, when we see a stop sign partially covered with snow and decide that it is indeed a stop sign, it is because, despite the snow, we are able easily to verify that the image matches a conceptual model of a stop sign with its properties (size, color, and location). However, a neural network must be trained to recognize stop signs in all possible weather conditions. This explains the difference between humans and neural networks in their speed of knowledge acquisition.

Furthermore, humans outperform machines in that they can effectively manage rare events and emergent situations. They can be creative/inventive and are not limited to managing a predefined number of goals.

In summary, for machines to match human situational awareness, they must be able to combine concrete sensory information with an extremely rich body of symbolic knowledge. The challenge is therefore to develop "self-learning systems" capable of progressively building semantic models of their environment by combining learning and reasoning techniques. This is probably the most difficult problem to solve, as shown by the poor progress made so far in the semantic analysis of natural languages.

5.3 Regulations and Ethical Issues

Trustworthiness is obviously a technical concept, but it also has a subjective and social dimension. Since we all live in one modern society or another, we cannot ignore the role of institutions that contribute directly or indirectly to shaping public perceptions on what is true, right, safe, and so on. The certification of critical systems has mostly remained the prerogative of independent agencies according to well-founded standards that require conclusive evidence that is based on models [61].

The acceptance of autonomous systems will depend on our decisions about when to trust them and when not to. Making these choices wisely depends on two factors. The first is our ability to define standards and regulations for autonomous systems based on robust and transparent evaluation criteria. In our view, the current trend toward self-regulation and self-certification should be a temporary stopgap measure rather than a permanent answer to the quest for trustworthiness. The development of the new standards will depend on the evolution of the state of the art and the willingness of authorities to exercise effective control for the protection of users.

The second factor is a heightened social consciousness and an ever-expanding sense of political responsibility. In general, public opinion is becoming less forgiving toward system failures than toward human errors, such as accidents caused by an autonomous car versus accidents caused by a human driver. Even if autonomous systems can eventually be made as trustworthy as humans, their acceptance to perform highly critical tasks will always be questioned.

It would be good to apply the precautionary principle that already underlies European Union laws and regulations: When computers are part of critical decision-making processes, we must ensure that their judgment is safe and fair [62]. Such a principle should be enshrined in the laws and regulations governing the development and deployment of autonomous systems.

The ethical question raised by the unregulated use of autonomous systems is whether we accept that critical decision-making processes rely on machine-generated knowledge that allows predictability without understanding. We believe that the threat is not that computer intelligence will come to surpasses human intelligence and that computers could take over human societies by plotting. The real danger comes from the massive replacement of responsible and accountable human operators in critical decision-making processes by computing devices and software.

Let us hope that we will not buckle under the pressure of economic interests and, based on doubtful performance benefits, will be tempted to grant decision-making power to autonomous systems without rigorous and strictly founded guarantees.

5.4 Toward a New Scientific and Engineering Foundation

We have explained that the autonomy vision involves important systems engineering issues that are not directly related to achieving intelligent behavior. Furthermore, monolithic end-to-end solutions based on artificial intelligence are not likely to be accepted, because they preclude trustworthiness guarantees by taking an explanation-lacking, verification-resisting approach. Instead, hybrid approaches are more viable, because they can leverage a robust body of knowledge for safe and effective decision-making combined with the effectiveness of AI-based techniques.

Since we have claimed that overall system validation is only possible through simulation and testing, it is important that we develop a sound theoretical and methodological framework for trustworthiness assessment. This framework will be empirical and similar to those used for knowledge development in physical sciences, although there are important differences between the physical world and the virtual world of system models.

Will the model-driven paradigm become obsolete due to the increasing diversity and complexity of autonomous systems? It is likely that we will not be able to apply rigorous development techniques to achieve the same level of confidence as for airplanes, e.g., 10^{-9} failures per hour of flight. In practical terms, this means that we should probably limit our ambitions for total autonomy and seek symbiotic autonomy schemes [63] that mark an appropriate division of labor between man and machine. Unless we devise new paradigms for system development, such as the use of correct-by-construction techniques, which would provide confidence guarantees without resorting to explicit system modeling.

Another crucial problem that we encounter in different contexts is to bridge the gap between symbolic knowledge and concrete information. In particular, this arises when we try to link symbolic models of the environment to concrete models obtained from the analysis of sensory information.

This problem motivates many works on explainable AI. The idea is to extract from a neural network mathematical models explaining its behavior. This is theoretically possible for a neural network, given its structure and the mathematical function that characterizes the input/output behavior of its nodes. In particular, for feed-forward networks this is achieved by propagating the input values along each layer to compute the output values. It is clear that the difficulty of this propagation depends on the type of activation function of the nodes.

Promising results for ReLU networks, for example, References [64, 65], pave the way for formal verification as well as robustness and sensitivity analysis. If the inherent complexity issues can be overcome, then we can expect better integration of data-based and model-based approaches in hybrid design flows.

Addressing the challenge of autonomy would be a major step in bridging the gap between machine and human intelligence. To achieve this, it is not enough to combine existing results from autonomous computing, adaptive systems, and autonomous agents. It will require a new scientific and technical foundation, which we have termed Autonomics, and which will no doubt require time and effort to build.

REFERENCES

[1] David Harel, Assaf Marron, and Joseph Sifakis. 2022. Creating a foundation for next-generation autonomous systems. *IEEE Des. Test* 39, 1 (2022), 49–56.

[2] D. Harel, Assaf Marron, and J. Sifakis. 2020. Autonomics: In search of a foundation for next generation autonomous systems. *Proc. Natl. Acad. Sci. U.S.A.* 117, 30 (2020), 17491–17498.

[3] Autonomic Computing. 2006. An architectural blueprint for autonomic computing. *IBM White Paper* 31, (2006), 1–6. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0e99837d9b1e70bb35d516e32ecfc345cd30e795.
[4] Michael I. Jordan. Artificial Intelligence—The Revolution Hasn't Happened Yet. Retrieved from https://hdsr.mitpress.mit.edu/pub/wot7mkc1/release/9.
[5] David Harel and Amir Pnueli. 1985. *On the Eevelopment of Reactive Systems, Logics and Models of Concurrent Systems* (K. R. Apt, Ed.). NATO ASI Series, F-13, Springer-Verlag, New York, 477–498.
[6] Joseph Sifakis. 2018. Autonomous systems an architectural characterization. arXiv:1811.10277. Retrieved from https://arxiv.org/abs/1811.10277.
[7] S. Efroni, D. Harel, and I. R. Cohen. 2005. Reactive animation: Realistic modeling of complex dynamic systems. *Computer* 38, 1, (2005), 38–47. DOI: 10.1109/MC.2005.31
[8] Simon Bliudze, Sébastien Furic, Joseph Sifakis, and Antoine Viel. 2019. Antoine viel: Rigorous design of cyber-physical systems—Linking physicality and computation. *Softw. Syst. Model* 18, 3 (2019), 1613–1636.
[9] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. 2015. *Decidability of Parameterized Verification, Synthesis, Lectures on Distributed Computing Theory*. Morgan & Claypool.
[10] J. Sifakis. 2012. Rigorous system design. *Foundations and Trends in Electronic Design Automation* 6, 4 (2012), 293–362.
[11] ISO Online Browsing Platform. Road vehicles— Functional safety— Part9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso:26262:-9:ed-1:v1:en.
[12] Wikipedia. V-model. Retrieved from https://en.wikipedia.org/wiki/V-model.
[13] Agile Alliance. Agile 101. Retrieved from https://www.agilealliance.org/agile101/.
[14] Rajmohan Madhavan, Elena R. Messina, and James S. Albus. 2006. *Intelligent Vehicle Systems: A 4D/RCS Approach*. Nova Science.
[15] James S. Albus and Anthony J. Barbera. 2005. RCS: A cognitive architecture for intelligent multi-agent systems. *Annu. Rev. Contr.* 29, 1 (2005), 87–99.
[16] Simon Ulbrich, Andreas Reschka, Jens Rieken, Susanne Ernst, Gerrit Bagschik, Frank Dierkes, Marcus Nolte, and Markus Maurer. 2017. Towards a functional system architecture for automated vehicles, arXiv:1703.08557 [cs.SY]. Retrieved from https://arxiv.org/abs/1703.08557.
[17] Sara Dersten, Jakob Axelsson, and Joakim Fröberg. 2015. An analysis of a layered system architecture for autonomous construction vehicles. In *Proceedings of the Annual IEEE Systems Conference (SysCon'15)*. 582–588.
[18] Thomas Braud, Jordan Ivanchev, Corvin Deboeser, Alois C. Knoll, David Eckhoff, and Alberto L. Sangiovanni-Vincentelli. 2021. AVDM: A hierarchical command-and-control system architecture for cooperative autonomous vehicles in highways scenario using microscopic simulations. *Auton. Agents Multi Agent Syst.* 35, 1 (2021), 16.
[19] Jonathan Aldrich, David Garlan, Christian Kästner, Claire Le Goues, Anahita Mohseni-Kabir, Ivan Ruchkin, Selva Samuel, Bradley R. Schmerl, Christopher Steven Timperley, Manuela Veloso, Ian Voysey, Joydeep Biswas, Arjun Guha, Jarrett Holtz, Javier Cámara, and Pooyan Jamshidi. 2019. Model-based adaptation for robotics software. *IEEE Softw.* 36, 2 (2019), 83–90.
[20] VIRES Simulationstechnologie GmbH. 2006. OpenDRIVE Format Specification. Tech. Rep. V 1.4.
[21] ASAM e.V. 2020. ASAM OpenDRIVE—Open Dynamic Road Information for Vehicle Environment. Tech. Rep. V 1.6.0.
[22] J. Beetz and A. Borrmann. 2018. Benefits and limitations of linked data approaches for road modeling and data exchange. In *Proceedings of the 25th EG-ICE International Workshop Advanced Computing Strategies for Engineering, Lecture Notes in Computer Science*, Vol. 10864I. F. C. Smith and B.Domer (Eds.). Springer, 245–261.
[23] G. Bagschik, T. Menzel, and M. Maurer. 2018. Ontology based scene creation for the development of automated vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV'18)* IEEE, 1813–1820.
[24] F. Poggenhans, J. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr. 2018. Lanelet2: A high-definition map framework for the future of automated driving. In *Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC'18)*, W. Zhang, A. M. Bayen, J. J. S. Medina, and M. J. Barth (Eds.), 1672–1679.
[25] Marius Bozga and Joseph Sifakis. 2021. Specification and validation of autonomous driving systems: A multilevel semantic framework, arXiv:2109.06478 [cs.MA]. Retrieved from https://arxiv.org/abs/2109.06478.
[26] Jean-Claude Laprie. 1992. Dependability: Basic concepts and terminology. In *Dependable Computing and Fault-Tolerant Systems*. Springer, Berlin, (1992).
[27] George Apostolakis. 2004. How useful is quantitative risk assessment? *Risk Anal.* 24, 3 (2004).
[28] W. S. Lee, D. L. Grosh, F. A. Tillman, and C. H. Lie. 1985. Fault tree analysis, methods, and applications a review. *IEEE Trans. Reliabil.* R-34, 3 (1985).
[29] Asim Abdulkhaleq, Stefan Wagner, and Nancy Leveson. 2015. A comprehensive safety engineering approach for software-intensive systems based on STPA. arXiv:1612.03109 [cs.SE]. https://doi.org/10.48550/arXiv.1612.03109.
[30] Malcolm Wallace. 2005. Modular architectural representation and analysis of fault propagation and transformation. *Electr. Not. Theor. Comput. Sci.* 141 (2005), 53–71.
[31] NHTSA. 2007. Pre-crash scenario typology for crash avoidance research, DOT HS 810 767.

[32] A Zolghadri. 2012. Advanced model-based FDIR techniques for aerospace systems: Today challenges and opportunities. In *Progress in Aerospace Sciences*. Vol. 53, Elsevier, 18–29.
[33] Asim Abdulkhaleq, Daniel Lammering, Stefan Wagner, Jürgen Rôder, Norbert Balbierer, Ludwig Ramsauer, Thomas Rastec, and Hagen Boehmert. 2017. A systematic approach based on STPA for developing a dependable architecture for fully automated driving vehicles. *Proceedings of the 4th European STAMP Workshop* 179 (2017), 41–51.
[34] John D. Schierman, Michael D. DeVore, Nathan D. Richards, Neha Gandhi, Jared K. Cooper, and Kenneth R. Horneman. 2015. Runtime assurance framework development for highly adaptive flight control systems, Barron associates. AFRL-RQ-WP-TR-2016-0001Final Report. Stony Brook University.
[35] L. Sha. 2001. Using simplicity to control complexity. *IEEE Softw.* 18, 4 (2001), 20–28.
[36] J. R. Mayo, R. C. Armstrong, G. C. Hulette, M. Salloum, and A. M. Smith. 2018. Robust digital computation in the physical world. In *Cyber-Physical Systems Security*. Springer, 1–21. DOI: 10.1007/978-3-319-98935-8_1
[37] M. Althoff, S. Maierhofer, and C. Pek. 2021. Provably-correct and comfortable adaptive cruise control. *IEEE Trans. Intell. Vehic.* 6, 1 (2021).
[38] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.* 20, 4 (2011), 1–64. https://doi.org/10.1145/2000799.2000800
[39] Antonio Bucchiarone and Juan P. Galeotti. 2008. Dynamic software architectures verification using dynalloy. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 10 (2008). DOI: http://dx.doi.org/10.14279/tuj.eceasst.10.145
[40] WAYMO. Waymo reaches 5 million self-driven miles. February 27, 2018. Retrieved from https://blog.waymo.com/2019/08/waymo-reaches-5-million-self-driven.html.
[41] Y. Setty, I. R. Cohen, Y. Dor, and D. Harel. 2008. Four-dimensional realistic modeling of pancreatic organogenesis. *Proc. Natl. Acad. Sci. U.S.A.* 105, 51 (2008), 20374–20379.
[42] N. Bloch, G. Weiss, S. Szekely, and D. Harel. 2015. An interactive tool for animating biology, and its use in spatial and temporal modeling of a cancerous tumor and its microenvironment. *PLoS ONE* 10, 7 (2015), e0133484. DOI: 10.1371/journal.pone.0133484
[43] ASAM Open. 2020. Scenario—Dynamic content in driving simulation, UML Modeling Rules. *Tech. Rep.* V 1.0.0, ASAM e.V.
[44] W. Damm and D. Harel. 2001. LSCs: Breathing life into message sequence charts. *Form. Methods Syst. Des.* 19, 1 (2001), 45–80.
[45] W. Damm, S. Kemper, E. Môhlmann, T. Peikenkamp, and A. Rakow. 2018. Using traffic sequence charts for the development of HAVs. In *Proceedings of the European Congress on Embedded Real Time Systems (ERTS'18)*.
[46] D. Harel and R. Marelly. 2003. *Come, let's play: Scenario-based programming using lscs and the play-engine*. Springer-Verlag, Berlin.
[47] D. J. Fremont, E. Kim, Y. V. Pant, S. A. Seshia, A. Acharya, X. Bruso, P. Wells, S. Lemke, Q. Lu, and S. Mehta. 2020. Formal scenario-based testing of autonomous vehicles: From simulation ta the real world. In *Proceedings of the 23rd IEEE International Conference on Intelligent Transportation Systems (ITSC'20)*. IEEE, 1–8.
[48] D. J. Fremont, E. Kim, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia. 2020. Scenic: A language for scenario specification and data generation. arXiv:2010.06580. Retrieved from https: //arxiv.org/abs/2010.06580.
[49] Antoine El-Hokayem, Marius Bozga, and Joseph Sifakis. 2021. A temporal configuration logic for dynamic reconfigurable systems. *SAC'21*. ACM, 1419–1428.
[50] Klemens Esterle, Vincent Aravantinos, and Alois Knoll. 2019. From specifications to behavior: Maneuver verification in a semantic state space. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV'19)*. IEEE, 2140–2147.
[51] A. Rizaldi and M. Althoff. 2015. Formalising traffic rules for accountability of autonomous vehicles. In *Proceedings of the IEEE 18th International Conference on Intelligent Transportation Systems (ITSC'15)*. IEEE, 1658–1665.
[52] A. Rizaldi, J. Keinholz, M. Huber, J. Feldle, F. Immler, M. Althoff, E. Hilgendorf, and T. Nipkow. 2017. Formalising and monitoring traffic rules for autonomous vehicles in Isabelle/HOL. In *Proceedings of the 13th International Conference on Integrated Formal Methods (IFM'17), Lecture Notes in Computer Science*, Vol. 10510, N. Polikarpova and S. A. Schneider (Eds.). Springer, 50–66.
[53] A. Karimi and P. S. Duggirala. 2020. Formalizing traffic rules for uncontrolled intersections. In *Proceedings of the 11th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'20)*. IEEE, 41–50.
[54] Z. Q. Zhou and L. Sun. 2019. Metamorphic testing of driverless cars. *Commun. ACM* 62, 3 (2019), 61–67.
[55] Antoine El-Hokayem, Saddek Bensalem, and Marius Bozga. 2020. Joseph Sifakis: A layered implementation of DR-BIP supporting run-time monitoring and analysis. In *Proceedings of the International Conference on Software Engineering and Formal Methods (SEFM'20)*. 284–302.
[56] SAE International Releases Updated Visual Chart for Its "Levels of Driving Automation" Standard for Self-Driving Vehicles. Retrieved from https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles.
[57] M. Jerrold and A. Grochow. 2020. Taxonomy of automated assistants. *Commun. ACM* 63 (2020), 39–41.

[58] Fernando Galdon, Ashley Hall, and Stephen Jia Wang. 2020. Designing trust in highly automated virtual assistants: A taxonomy of levels of autonomy. In *Artificial Intelligence in Industry 4.0: A Collection of Innovative Research Case-studies.*

[59] Ernest Davis and Gary Marcus. 2015. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM* 58, 9 (2015), 92–103.

[60] NDTV Watch: Tesla Autopilot Feature Mistakes Moon For Yellow Traffic Light. Retrieved from July 27 2021 https://www.ndtv.com/offbeat/watch-tesla-autopilot-feature-mistakes-moon-for-yellow-trafficlight-2495804.

[61] P. G. Neumann. 2017. Trustworthiness and truthfulness are essential. *Commun.* ACM 60, 6 (2017), 26–28.

[62] Wikipedia. Precautionary principle. Retrieved from https://en.wikipedia.org/wiki/Precautionary_principle.

[63] Stuart Mason Dambrot, Derrick de Kerchove, Francesco Flammini, Witold Kinsner, Linda MacDonald Glenn, and Roberto Saracco. 2018. *IEEE Symbiotic Autonomous Systems White Paper ii.*

[64] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. arXiv:1702.01135v2 [cs.AI]. Retrieved from https://arxiv.org/abs/1702.01135v2.

[65] Nicola Franco, Tom Wollschläger, Nicholas Gao, Jeanette Miriam Lorenz, and Stephan Günnemann. 2022. Quantum robustness verification: A hybrid quantum-classical neural network certification algorithm. arXiv:2205.00900v1 [quant-ph]. Retrieved from https:arxiv.org/abs/2205.00900v1.

Received 7 April 2022; revised 20 June 2022; accepted 20 June 2022