SIEMENS
Ingenuity for life

Siemens Digital Industries Software

Simcenter Prescan360

Providing smart verification and validation for autonomous vehicles

Executive summary

The evolution from a human-driven to a predominantly driverless automotive ecosystem is gradually, but inevitably taking place. As technology becomes increasingly pervasive and complex, automobiles are being transformed into sophisticated machines that can sense, perceive and make decisions autonomously in real time. The degree of complexity of software and control systems required to realize these autonomous machines will rise orders of magnitude beyond human comprehension of logical complexity. As these machines are authorized to make decisions on their own, manufacturers will have to verify the integrity and authenticity of the software and hardware that are wired into them. Before putting these complex systems into practice, it is therefore important to simulate how they will perform under different scenarios.

Vinay Ramanath, Siemens Corporate Research and Technology
and Matthieu Worm, Siemens Digital Industries Software

Abstract

It is critical to gain confidence in the functional safety performance of these systems and ensure they behave as intended by a defined set of rules. This white paper provides insight into Siemens Digital Industries Software's approach to verifying and validating high-integrity systems operating in autonomous environments across the entire system of systems. This starts with the design of the chips and goes through system and vehicle development up to the city level, where autonomous driving vehicles become part of mobility solutions.

With the introduction of Simcenter™ Prescan™ software 360, Siemens introduces an off-the-shelf solution for validating and verifying autonomous driving vehicles on a mass scale. With the integration of a series of core products from within the Siemens portfolio, a process and toolchain has emerged that is being enriched with knowledge of formal verification, design exploration and design of experiment (DoE) methodologies from our corporate technology teams.

Siemens introduces an off-the-shelf solution for validating and verifying autonomous driving vehicles on a mass scale.

Coverage-driven AV performance verification

In this white paper we propose a user-friendly approach for defining and automatically translating requirements into a temporal logic language. Then we analyze the inconsistency in the requirements via model checker tools and define the region of interest in scenario planning for all actors. Finally, a down select of parameters enables an automated, constrained algorithm to provide feasible scenario combinations given correlated input parameters. This generates a formal verification algorithm that automatically identifies edge cases or counter cases and thus enables trapping the bugs upfront. A probabilistic behavior modeling framework creates a network of events to predict the likelihood of possible outcomes.

The problem definitions can be broadly classified into four areas:

1. Requirements specification and modeling

The requirements specification needs to define the rules of the autonomous systems and should have a clear description of how agents should behave in different scenarios and be clear about the in-and-out scope of the simulation envelope. Modeling the requirements requires the use of formalism to resolve ambiguity and identify contradictions in requirements.

2. Smart testing scenarios modeling

The number of scenarios in autonomous vehicle (AV) simulation testing will grow exponentially with an increase in the number of scenario variables. Many of these scenarios will be infeasible due to the assumption of independent variables. It is therefore critical to focus on testing the right scenarios by considering realistic correlations of variables that exist in practice. For instance, a scenario in which there is heavy snow and hot temperatures is unlikely to occur. Such variable correlations must be defined for the purpose of experimental screening so only the right and feasible scenarios get simulated and tested. In the next step, a falsification approach is required to minimize the number of irrelevant scenarios, driving efficiency in the massive validation and verification execution plans.

3. Behavioral modeling of agents

Beyond simulation-based testing, it is critical to establish a probabilistic network of agent behavior in an autonomous environment to predict key events and thereby activate the alert system for the vehicle user. This is needed to verify the risk of accidents in real time and alert/take mitigation actions for accident avoidance. Bayesian modeling is one such probabilistic network that can be employed to create a dynamic network, which can be updated based on real-time information from perception-based devices (sensors, radars, lidars, etc.).

4. Reliability and security modeling

An AV can perceive its environment, navigate and maneuver without human actions. The AVs will have a high degree of acceptability when the security risks of automation are enumerated and evaluated. It is therefore vital to recognize threats, classify them and develop protection strategies for AVs. The goal is to develop a security model to quantify the risk and assess the likelihood of threats on AV components. Further, we should look at learning from failures observed during testing at the lab and make them part of the future test strategy on a continuous basis, typically in a distributed, multi-site scenario.

Simcenter Prescan360

Figure 1 shows the high-level architecture of Simcenter Prescan360 for the first generation of coverage-driven AV performance verification. It depicts design/process orchestration engine (HEEDS™ software) and the virtual simulation setup for autonomous systems (Simcenter Prescan), providing technical details about the process, and highlighting the possible process steps to be covered for realizing coverage-driven AV performance verification. This view is to be completed with the usage of requirements management tools, such as Polarion. Such tools will be interfaced in an increasingly automated manner in the future, to bring closed-loop, systematic virtual verification through simulation.

Requirements specification and modeling

In a traditional systems-engineering lifecycle, the system is initially specified. This specification is then cascaded with increasing levels of technical details. Following that, the validation and verification of each element that has been specified is tested to see if it has been achieved. Extending the same principles to a fully integrated AV, the autonomous system is not the highest level of system that we must deal with. Instead, the higher level of the transport and operational systems need to be considered and validated to emerge with a seamless integration productivity [1].

Additionally, in terms of validation and verification, scenario-based simulation and testing fundamentally require mapping test cases into requirements. For example, requirements might involve control and stabilizing of speed: Test cases should reflect the conditions in which this maneuverability might be challenging, like weather conditions, tire mechanics, etc. Hence, a scenario-based simulation is used to map specific requirements into a nucleus of test cases, thereby making the requirements representative within an operational design domain. Therefore, in addition to the traditional set of user requirements, the operational design domain can exhibit the requirements the autonomous system needs to exhibit in different scenarios. This journey then results in a massive set of requirements and scenarios that leads to the necessity of containing or limiting it. This necessity poses an interesting challenge for defining validation and verification requirements.

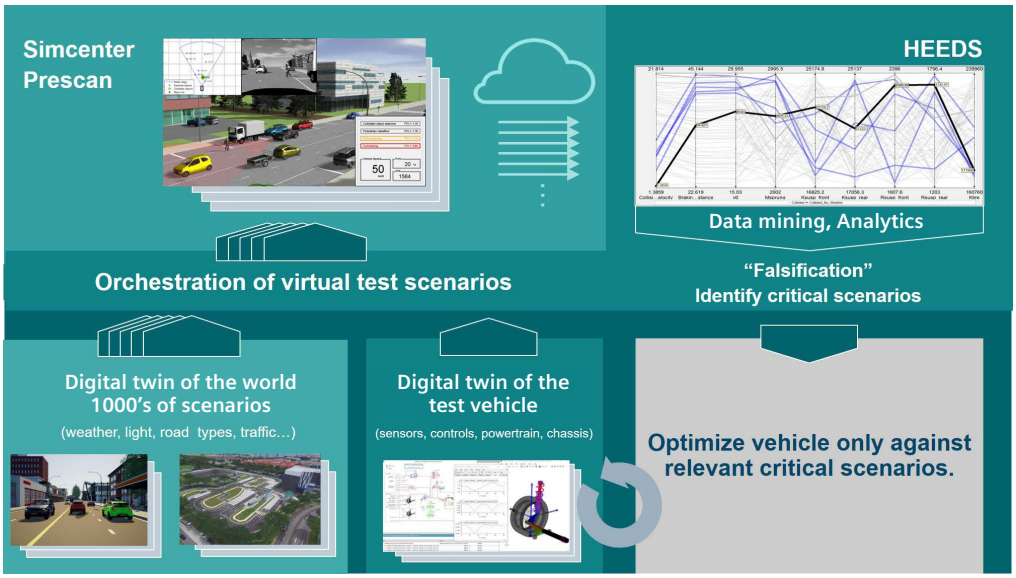

Figure 1. Introducing Simcenter Prescan360.

Formal verification approaches

There have been several accidents involving autonomous vehicles and concerns regarding their safety. This has underlined the importance of rigorous and more formal verification and validation of autonomous systems. Just simulating millions of miles is not adequate to provide enough confidence regarding the safety of autonomous vehicles.

The broad process steps for autonomous vehicle validation and verification are:

1. Establishing standards for requirements and specification documents
2. Formalizing functional requirements and safety requirements
3. Automatic translation of a simulation model to a formal model
4. Temporal logic specification for formal description of the system requirements (the property to be checked)
5. Execution of a model checker to ensure the formal model satisfies the properties and generates counterexamples

Formal methods, especially model checking, have been used successfully in the area of software program verification [2]. One possible way forward for verifying autonomous vehicles is to extend and adapt formal verification techniques for autonomous systems [3,4]. This involves two key challenges: formally modeling the requirements in an extension of temporal logic suitable for AVs such as metric temporal logic, signal temporal logic, spatial temporal logic, timed temporal logic, etc., and deriving a formal automation of various simulation models such as environment models, vehicle models, etc.

A complementary approach is to automatically generate edge test cases using standard optimization techniques (for example, stochastic optimization with adaptive restarts) guided by formal requirements in an appropriate temporal logic in order to find falsifying system behaviors [5].

A high-level sketch of how such formal approaches could be employed is shown in figure 2.

As shown in figure 2, the requirements, both internal and external, are defined in Polarion REQUIREMENTS. The formal model toolbox will translate the requirements defined in natural language to a signal temporal logic formula. Further, the temporal logic is translated to a constraint specification syntax understood by Simcenter Prescan360. The formal model receives the simulation output from Simcenter Prescan360 and provides a decision. The decision outcome from the formal model will lead towards either a) a successful verification, b) counter example generation or c) edge case generation. The counterexample and edge case scenarios are looped back to the simulation platform for further processing and validation purposes.

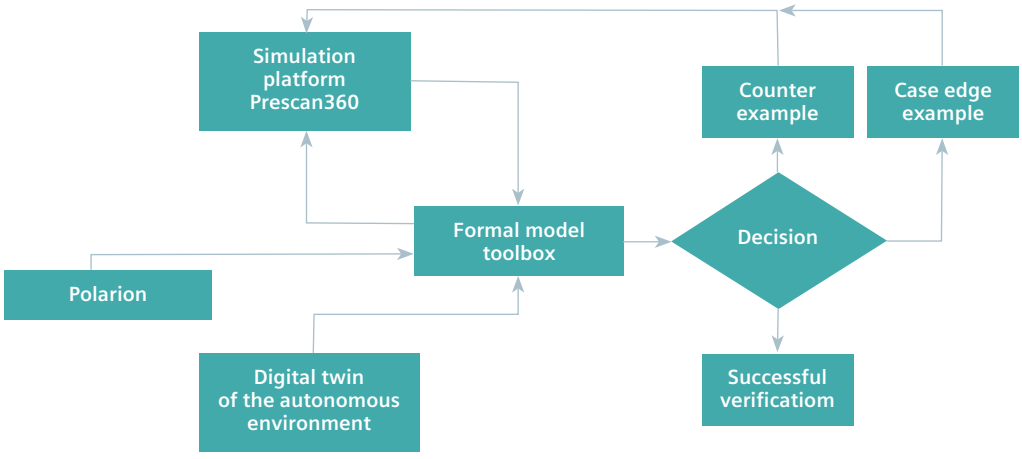

Figure 2. Process flow chart for formal elicitation and verification.

Scenario modeling

As described in figure 3, the extensive in-traffic test requirements for AV can only be satisfied by simulation. Simulation software like Simcenter Prescan and Siemens' Aimsun Auto can be used to simulate realistic road conditions with many other vehicles (including AVs) and actors (like pedestrians) on different roads and in different environments to validate system performance and safety. Further, other solutions such as Waymo Carcraft, PTV Vissim and Caliper TransModeler can be interfaced with Simcenter Prescan and Aimsun Auto. The edge and corner test cases can be generated and validated. The robustness of the system performance can only be tested by simulation since these edge and corner cases cannot be reproduced accurately in outdoor testing.

Accidents happen on the road every day. These accident scenarios can provide a valuable source of edge-case scenarios. It is important to understand the reason for the accident so the system can be upgraded based on the experience. With simulation we can recreate accident scenarios. For example, accident databases like German In-Depth Accident Study (GIDAS) and China In-Depth Accident Study (CIDAS) contain accident descriptions that can be imported into Simcenter Prescan.

Vehicle modeling

Preliminary system design involving many subsystems generally starts with conceptual design followed by design of subsystems and integration. Simulation of systems is inevitable to avoid high redesign cost before prototyping. Integrating subsystems with different complexity levels is needed in order to fine-tune and validate system parameters to meet the intended design requirements. Multi-physics system simulation software solutions like Simcenter Amesim™ software and MATLAB's Simulink environment can be exploited to get insights into the systems response before hardware is available.

Subsystems like sensors, actuators, etc., and the associated algorithms can be validated and verified with simulation using hardware-in-the-loop (HiL), software-in-the-loop (SiL) and model-in- the-loop (MiL). In this case, software like Simcenter Amesim and Simcenter Tyre software can be used. Scenarios in Simcenter Prescan can be looped in order to simulate and verify a variety of variants based on system parameters. For example, road-to-tire friction for a specific circumstance can be validated by a high-fidelity simulation of a representative model.

Figure 3. Simcenter Prescan.

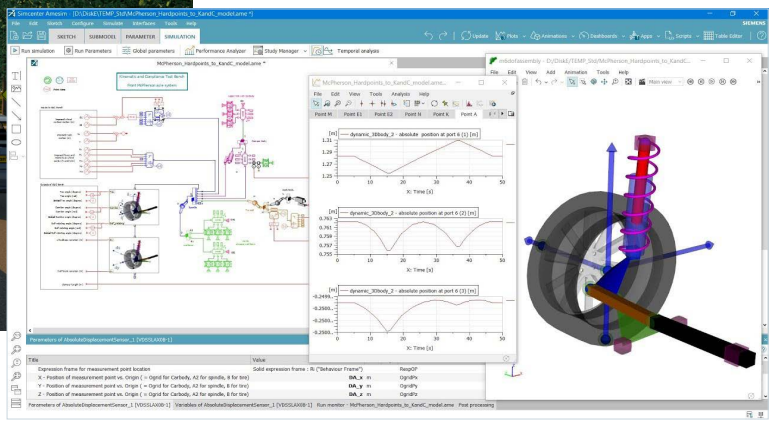

Figure 4. Simcenter Amesim.

Smart testing scenario planning

Quantifying coverage metrics and defining the boundaries for variable values sets the tone for smart test planning. Coverage metrics drawn from computer sampling can be employed; for example, to define the space filling of sampling, criterion based on inter sample electric force distribution, minimum/maximum distance, entropy, etc., are deployed for addressing the coverage quantification.

Smart test planning employs a machine-learning-based sampling algorithm with the ability to handle correlated inputs to generate the required feasibility samples. For example, the occurrence of the variable combination of "Snow Sleeting = heavy" and "Hot day temperature > 35 C" is most unlikely and needs to be constrained while constructing a testing scenario, along with other actors (or variables).

In order to model such dynamic scenarios using simulation tools, there are scene templates available in Simcenter Prescan. If there is a need for a detailed model to simulate behavior or response of the vehicle to the environment, actors or road conditions, an appropriate fidelity model in Simulink and Simcenter Amesim can be employed. At an abstract level, a physics-based empirical equation from first principles can be formulated in many contexts. One such elementary abstract example will be explained below.

The multi-fidelity, multidisciplinary and multi-objective process is constructed with HEEDS, which is a system-level integration software/platform. This platform provides tools and algorithms for exploring the parameter space of interest. In special applications such as edge-case testing, smart design space exploration methods can consider intercorrelated input parameters, thereby arriving at feasible design parameter boundaries.

In an example scenario, consider a subject vehicle (SV) moving with y speed detecting a lead vehicle (LV) in front in the same lane. The SV is supposed to apply brakes or change lanes. The braking distance must be less than the safe stopping sight distance (SSD) and depends on the speed of the SV and LV. The braking distance is the sum of the reaction distance and stopping distances. The reaction distance is a function of reaction time. This reaction time depends on road visibility and is influenced by weather and environmental conditions. Although the stopping distance is a function of speed, it is also influenced by tire-road interaction. The tire-road interaction is modeled using a coefficient of friction that is dependent on whether the road surface is dry, wet, snowy, etc. Further, weather conditions significantly influence road visibility. Hence, it is important to consider the interdependency between input parameters during design exploration. Otherwise, the simulated scenario would result in infeasible/imaginary solutions.

Table 1 shows the variable definitions and values for the demonstration problem.

Variable – weather	Variable – friction coefficient	Variable – visibility
Dry (=> 0)	0.9	Normal (=> 1)
Fog (=> 1)	0.55	Less (=> 0.8)
Light rains (=> 2)	0.4	Low (=> 0.7)
Heavy rains (=> 3)	0.3	Very low (=> 0.6)
Snow/ heavy snow (=> 4)	0.2	Remote (=> 0.5)

Table 1. Variable definitions and values for assessing the safe braking distance.

Referring to table 1, it is evident that weather, friction coefficient and visibility forms the three variables and each of them assumes five discrete values. There are a total of 125 combinations of experimental designs (five levels for each variable with a total of three variables -> 5^3 -> 125) that need to be simulated. However, the key question is whether some of the variable combinations are infeasible and does it warrant them being excluded from experimental screening? Experimental combinations "dry weather -> low friction coefficients -> high visibility." "fog -> low friction coefficients -> high visibility," "snow/ heavy snow condition -> high friction coefficients -> high visibility" are unlikely to occur and hence should be considered as infeasible from a simulation perspective. It is important to automatically identify such infeasible combinations from the experimental setup so only the right and feasible testing scenarios are tested and simulated.

Figure 5 shows the design space exploration for 100 samples for methods without constraint pruning. The feasible design space is a small fraction of the complete space; hence, a randomized sampling followed by constraint pruning will make it difficult to identify the feasible design space boundary.

To perform efficient sampling in constrained spaces, a proprietary algorithm integrated in HEEDS for the test automation with a capability of constraint pruning on input parameters is employed in the demonstration example.

To satisfy feasibility conditions, the constraints are formulated as follows in the machine-learning-enabled smart sampling approach:

1. If weather is fog, friction > 0.5 and visibility < 0.2 (a scenario in which foggy conditions, better traction and low visibility are observed).
2. If weather is dry, friction > 0.5 and visibility > 0.8 (a scenario in which dry conditions, better traction and high visibility are observed).
3. If weather is snow/heavy snow, friction <= 0.2 and visibility around 0.5 (a scenario in which snow conditions, least traction and medium visibility are observed).
4. If weather is heavy rain, friction between 0.2, 0.4 and visibility around 0.6 (a scenario in which heavy rain conditions, lower traction and medium visibility are observed).

The smart sampling algorithm consumes these constraints and effectively generates the required number of feasible scenarios as shown in figure 6.

It is evident from figure 6 the smart sampling algorithm can identify the feasible design space accurately and the required number of feasible samples can get generated, satisfying the inter parameter constraints.

The smart sampling approach is valuable in an ADAS and AV controller validation and verification context but could be instrumental as well in the design stage of controllers. For example, Iterative Learning Control (ILC) techniques require system repeatability, that is, a system is required to follow an identical or similar trajectory. As depicted by S. Tong, during the development of ILC algorithms disturbances need to be added to grow robustness [6]. Having an automation environment like Simcenter Prescan360 makes this process much more efficient.

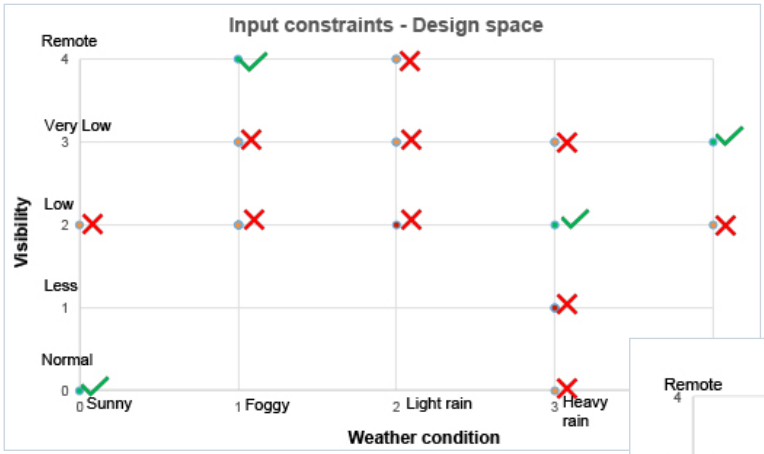

Figure 5. Design space exploration without constraint pruning.

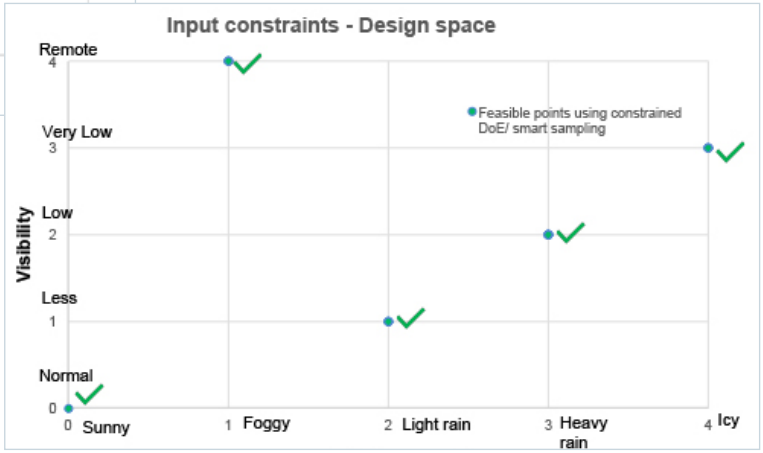

Figure 6. Design space exploration with smart sampling.

Behavioral modeling of agents

As has been mentioned, at the highest level of automation, the autonomous vehicle is expected to perform the driving tasks of an alert and nonaggressive driver. There is an increased interest in the automobile community to model the behaviors of agents using artificial intelligence (AI). There are several approaches to adding intelligence to a process, function or agent in an algorithm. In automotive engineering, Bayesian intelligence offers the promise of enabling the system to learn, adapt and improve and thereby to become human-like.

Beyond discrete testing methods, probabilistic modeling with a Bayesian approach that predicts the probability of events given a scenario can be employed. An instance of employing Bayesian rules for modeling the functions of car-following and lane-change models is described. The Bayesian methodology has been used successfully as the foundation for system design and decision analysis in challenging situations in which uncertain states of nature are encountered and learning opportunities are available to refine knowledge of uncertain factors. Bayesian AI can play the following role in the context of car-following and lane-change scenarios:

1. To complement or replace the human driver, Bayesian AI must reason in spite of uncertainties in the driving environment.
2. In the driver assistance role, AI will need to reason whether the driver is alert and is driving in a nonaggressive manner. If not, AI is in charge in the automated mode and the driver is not in the loop, so AI has to process available data/information, estimate probabilities and identify the most favorable decision in driving.
3. The Bayesian AI can update probabilities and wait for new information to become available for modifying probabilities to decide. That means we should be able to blend old information with new information with Bayesian math, enabling decision-making in a dynamic and uncertain driving environment.

Some of the scenario examples that can be answered by probabilistic modeling of events are:

1. What is the probability of a frontal crash given the leading vehicle suddenly applies the brakes and the follower vehicle takes time to respond to the event?
2. What is the risk of a lateral crash if a leading vehicle accepts a lane change (lane 1 -> lane 2) while the following vehicle, which is moving at a higher speed in lane 2, takes time to respond to the event before braking?

Reliability and security modeling

This section provides a description of defining metrics for AV robustness and process to ensure the trustworthiness of AV.

One way to provide robustness guarantees is with the issuance of partial certificates (called μ certificates from here on) for functionality in which there is a high degree of confidence. It is also possible to have a hierarchy of certificates like the SAE Autonomous Driving Levels indicating the level of trust and reliability in the system being tested.

The mission of autonomous vehicles is to navigate safely, securely and reliably from an origin to a destination, which are chosen by the user. In order to fulfill the mission, navigation decisions are automated by acquiring four classes of information, namely: (a) environmental perception information from perception sensors such as camera, radar and lidar, (b) relative location information from machine-to-machine communication sensors such as Global Positioning System (GPS), vehicle-to-vehicle (V2V) communications and vehicle-to-infrastructure (V2I) communications, (c) path planning information from networked services such as Google Maps or other mapping services and (d) synchronization information from networked services like Network Time Protocol and Precision Time Protocol. Autonomous vehicle decision subsystems must be used to fuse the four classes of acquired information in real-time in order to generate driving instructions for the vehicle control system.

The mission of autonomous vehicles could be jeopardized by falsifying or denying the four classes of information. Environmental perception information can be falsified or delayed (denied) by jamming the corresponding sensors. Petit [7] provides a detailed account on how lidar and camera sensors can be jammed. Similarly, relative location information can also be falsified or delayed (denied) by spoofing attacks [8,9]. Path planning information can be made unavailable by jamming telecommunication signals used by the autonomous vehicle to consume information from remote services. Attacks on synchronization [10,11] can attack safety-critical coordination functions.

In order to gain confidence about the safety, security and reliability of autonomous vehicles, the safety and liveness properties of the distributed system [12, chapter 7], of which the autonomous vehicle is a part, needs to be established. Modeling and simulation provides efficient techniques for proving the safety and liveness properties in (a) normal operations – liveness, (b) operations when some components are faulty – reliable safety and (c) operations in the presence of adversarial actions – secure safety.

In order to realize safety in a secure manner, adversarial actions need to be modeled so the resultant simulation can help in discovering the consequence of falsifying or denying any of the four classes of information to autonomous vehicles. There are two approaches to simulating cybersecurity, namely design simulation and operations simulation.

In design simulation, adversarial actions are modeled as discrete events for corrupting the four classes of information available to autonomous vehicles. Any discrete event simulation tool that can interface with the autonomous driving system simulation can be used. The consequence of such adversarial actions can be explored by simulating the model. Design simulation is useful for creating robust design specifications and security certification. For example, the NS3 network simulator [13] can be used to simulate attacks on relative location, path planning and synchronization.

Operations tend to be HiL simulations that aim to subject an embedded system to adversarial actions while containing the adversarial actions in a sandbox. Such simulations are useful for operations design and training purposes. The SANS Netwars simulator [14] is an example for this approach.

Cyber-physical system simulation for autonomous vehicles would include the use of an array of simulation tools and techniques. It will encompass the development, testing and marketing of a new class of simulation products to assist in the secure design and development of components, subsystems and systems for autonomous vehicles.

Conclusion

The introduction of Simcenter Prescan360 for massive virtual validation and verification of ADAS and AV technology not only marks the launch of an off-the-shelf engineering environment for autonomous vehicles. Perhaps even more importantly, it is the starting point for including a smart approach. The step from testing large numbers of miles to coverage-driven validation and verification can now be set.

References

1. Edwards, Tim (2019). Verification and validation challenges for connected and autonomous vehicles, verification futures.
2. Jhala, Ranjit; Majumdar, Rupak (2009). Software model checking. ACM Comput. Surv. 41, 4, Article 21.
3. Khosrowjerdi, Hojat and Meinke, Karl (2018). Learning-based testing for autonomous systems using spatial and temporal requirements, in proceedings of the 1st International Workshop on Machine Learning and Software Engineering in Symbiosis.
4. Kamali, Maryam; Dennis, Louise; McAree, Owen; Fisher, Michael; Veres, Sandor (2017). Formal verification of autonomous vehicle platooning, Science of Computer Programming, Volume 148.
5. Tuncali, Cumhur; Fainekos, Georgios; Ito, Hisahiro; Kapinski, James (2018). Simulation-based adversarial test generation for autonomous vehicles with machine learning components, IEEE Intelligent Vehicles Symposium (IV).
6. S. Tong, A. Bhave, T. Geluk, H. Van der Auweraer, 2019, Autonomous Driving Control in Safety Critical Scenarios, Proc. JSAE Annual Spring Congress 2019, May 22-24, Yokohama (JP).
7. Petit, Jonathan (2019). Self-driving and connected cars: fooling sensors and tracking drivers.
8. Kaspersky AG (2019). Is it possible to guard against GPS attacks?
9. Amoozadeh M.; Raghuramu A., Chuah C.; Ghosal D.; Zhang H.; Rowe J.; Levitt K. (2019). Security Vulnerabilities of Connected Vehicles Streams and their Impact on Cooperative Driving.
10. Hampson, Michelle (2019). It's Surprisingly Easy to Hack the Precision Time Protocol.
11. Malhotra A.; Cohen I.; Brakke E.; Goldberg S. (2019). Attacking the Network Time Protocol.
12. Magee J.; Kramer J. (2019). Concurrency: State Models & Java Programs.
13. NS-3 (2019), NS-3: Network Simulator (2019).
14. SANS (2019), Netwars simulator.

Siemens Digital Industries Software

Headquarters
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 972 987 3000

Americas
Granite Park One
5800 Granite Parkway
Suite 600
Plano, TX 75024
USA
+1 314 264 8499

Europe
Stephenson House
Sir William Siemens Square
Frimley, Camberley
Surrey, GU16 8QD
+44 (0) 1276 413200

Asia-Pacific
Unit 901-902, 9/F
Tower B, Manulife Financial Centre
223-231 Wai Yip Street, Kwun Tong
Kowloon, Hong Kong
+852 2230 3333

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Our solutions help companies of all sizes create and leverage digital twins that provide organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram.
Siemens Digital Industries Software –
Where today meets tomorrow.

siemens.com/software

78898-C3 1/20 A