Hierarchical Assessment of Safety Requirements for Configurations of Autonomous Driving Systems

Yixing Luo*†, Xiao-Yi Zhang‡, Paolo Arcaini‡, Zhi Jin*†, Haiyan Zhao*†, Linjuan Zhang*†, Fuyuki Ishikawa‡,
*Key Lab. of High-Confidence Software Technologies (Peking University), Ministry of Education, Beijing, China
†School of Computer Science, Peking University, Beijing, China
‡National Institute of Informatics, Tokyo, Japan
Email: {yixingluo, zhijin, zhhy.sei, genesis}@pku.edu.cn {xiaoyi, arcaini, f-ishikawa}@nii.ac.jp

***Abstract*—Autonomous Driving Systems (ADSs) are complex systems that must satisfy multiple safety requirements. In particular cases, all the requirements cannot be satisfied at the same time, and the control software of the ADS must make trade-offs among their satisfaction. Usually, the trading-offs in the decision-making process are configurable; different configuration options can affect driving behaviors, satisfying or violating requirements at different degrees. Therefore, it is highly important to know whether a configuration can guarantee a safe drive or not, i.e., whether it leads to the requirement violations that exceed the allowable range or not. However, there is currently no approach to systematically assess the safety of ADS configurations from the perspective of requirements violations. To bridge this gap, this paper proposes a "Hierarchical Safety Assessment" approach (`HSA`) that is able to quantitatively analyze violation severity of safety requirements and distinguish safer ADS configurations based on the requirements violations comparison done in a hierarchical way by following requirements importance. We apply `HSA` to an industrial ADS under six traffic situations. Evaluation results show that `HSA` is effective in distinguishing safer configurations and provides useful feedback to ADS engineers to reconfigure the ADS in a better way.**

***Index Terms*—Requirements Violation Analysis, Safety Assessment, Autonomous Driving Systems, Configuration**

I. Introduction

Autonomous Driving Systems (ADSs), which control the physical vehicle using sensors and actuators with intelligence provided by software and data, are transforming the domain of transportation. An important issue through this transformation is safety assurance [1], as failures of ADSs could result in accidents that cause damages to the environment, financial losses, injury to people, and loss of lives.

Due to the close interaction with highly open and dynamic environments, the ADS is expected to meet multiple safety requirements both from system-centric and environment-centric perspectives [2] to protect the system and the environment from harm. Czarnecki [3] identifies five categories of safety requirements on driving behaviors for ADS-operated vehicles: maintaining vehicle stability, assuring cleared distance ahead, keeping minimum separation, compliance to traffic regulations, and adherence to best practices. Violations of safety requirements are the major source of accidents, and numerous crashes have occurred during the operation of ADSs [4], [5]. Thus, identification and analysis of violations of safety requirements are important for the safety of ADSs [6].

As one of the key components in the ADS, the control software [7] is responsible for generating optimal trajectories to meet the aforementioned safety requirements. Indeed, satisfying all the requirements at the same time may not be possible, as unexpected events may happen in highly open and dynamic environments, e.g., intrusion of a hidden traffic participant or weather disturbances [8]. Sometimes, the control software of the ADS has to make trade-offs among different requirements satisfactions. The trading-offs in the decision-making process are determined by some *configuration options* of the ADS; using different *configurations* (i.e., specific values for the configuration options) leads to the selection of different trajectories to track and to different results of requirements violations (e.g., more serious violations or violation of more requirements) [9]. Different driving situations may require different configurations to guarantee a safe drive.

The identification of the best configuration for obtaining a safe ADS in given traffic situations can be performed either at runtime or in the development process. Runtime reconfiguration [7], [10], to a certain extent, enables the autonomous vehicle with more flexibility in response to unexpected events; however, it is difficult to find a safe configuration at runtime within limited time. Therefore, it is better to identify a set of safe configurations with respect to different requirements at design time, which can form a knowledge base of configurations for use at runtime. To this aim, approaches for configuring ADS in simulated environments [9], [11] have been proposed to find alternative configurations for safe driving.

However, there are deficiencies in these works on the assessment of ADS safety across different configurations. First, existing research only finds the configurations that are shown to be better on ad-hoc scenarios [11], but it is not clear whether they are suitable also for other (similar) scenarios in given traffic situations. Second, the alternative configurations only consider a single safety requirement, and most of the time is

This work is supported in part by the National Natural Science Foundation of China under Grant No. 61620106007 and 61751210. Zhi Jin is the corresponding author. X. Zhang, P. Arcaini, and F. Ishikawa are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, and MIRAI Engineerable AI Project (No. JPMJMI20B8), JST. We thank our industry partner Mazda for providing the software used in our work and discussing principles in testing and improving complex real-world automotive systems. The provided software is a prototype constructed for the purpose of evaluating new testing techniques, and its quality has no relation with the quality of Mazda products.

the safe distance, while other safety-related requirements like compliance to traffic regulations (e.g., lane-keeping) whose violations may also lead to accidents are not considered.

Instead, it is important to have a systematic approach to measure the violation of safety requirements of ADS under different configurations in given traffic situations. However, this is a challenging problem. The first challenge is how to measure the violation of safety requirements in specific scenarios, as the severity of accidents depends on the states of both the autonomous vehicle and involved road users. The second challenge is how to compare the safety of ADS configurations with respect to multiple safety requirements.

To address the preceding challenges, our work is based on the following principles. First, besides evaluating the violation/satisfaction of safety requirements in specific scenarios, the requirements violation *severity* should be quantitatively measured from the following three aspects: to what degree the requirement is violated, how long the violation lasts, and how many times the violation happens. Second, based on the above quantitative evaluation results, the distinguished safer ADS configurations should first focus on the satisfaction of safety requirements with higher *importance*, as the violations of these requirements may lead to more serious accidents for the ADS.

Based on these principles, in this paper, we propose a hierarchical assessment approach, `HSA`, that can effectively distinguish safer ADS configurations driven by the different *importance* of safety requirements. The approach requires that the safety requirements are prioritized by *importance level*. Then, we propose a quantitative technique, *requirements violation analysis* (`RVA`), to analyze the severity and importance level of violated requirements according to the driving behaviors of the ADS in a specific scenario. `RVA` produces two types of results, *requirements violation severity* and *requirements violation mode* (indicating a specific combination of requirements violation). Driven by the importance level of safety requirements, we propose a hierarchical requirements violation comparison (`RVC`) which incrementally checks whether, for scenarios in a given traffic situation, one configuration is better than another one in satisfying the requirements, considered from the highest to the lowest importance level.

This paper makes the following main contributions:

- A quantitative model, `RVA`, to analyze violations of safety requirements in specific scenarios for the differently configured ADS.
- A hierarchical comparison technique, `RVC`, to compare the analysis results of `RVA` for the ADS under different pairs of configurations in a systematic way.
- An assessment approach, *Hierarchical Safety Assessment* (`HSA`), which combines the results of `RVA` and `RVC` to rank the ADS configurations from the perspective of violations of the safety requirements.
- Experiments showing the effectiveness of `HSA` at distinguishing safer configurations for the ADS (RQ1, RQ2), and a demonstration of the possible usage of the results in an industrial context when configuring the ADS (RQ3).

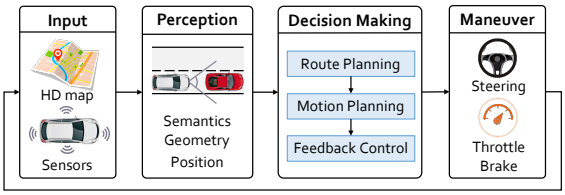


Fig. 1: Overview of an Autonomous Driving System (ADS)

II. Motivation

Fig. 1 shows an overview of an Autonomous Driving System (ADS) which is composed of a pipeline of individual modules, linking sensor inputs to motor outputs. The *ego vehicle* is the autonomous vehicle equipped with the ADS that runs iteratively at regular time steps. The environment is perceived by the high-definition map and sensor inputs at runtime. The *perception* module is responsible for three important tasks: estimate the semantics of the surrounding scene, understand its geometry, and estimate the position of the ego vehicle itself [12].

As the core of the ADS, the *decision-making* process in the control software is decomposed into the route planning, motion planning, and feedback control [13]. The route planning achieves the navigation mission, i.e., a route from an initial position to the destination is planned through the road network. This is followed by motion planning, which decides an optimal and continuous trajectory for the ego vehicle to follow. In the motion planning, safety requirements that are related to different stakeholders, such as passengers, pedestrians, and the authorities, are considered. In the feedback control, errors (if any) in the execution of the planned motion (e.g., steering, throttle, and brake commands) are reactively corrected.

As a case study, we use an ADS with an optimization-based Motion Planner, `MP`, provided by our industry partner. `MP` can run in a simulator, as shown in Fig. 2. Besides the navigation mission, `MP` considers seven safety requirements as follows:

- R_1: avoid impossible steering angles.
- R_2: keep a safe distance from other objects.
- R_3: keep the velocity below the speed limit.
- R_4: stay in the correct lane.
- R_5: avoid too much acceleration.
- R_6: avoid too much deceleration.
- R_7: avoid too much lateral acceleration.

In the motion planning, `MP` generates an optimal trajectory by listing and scoring a set of possible trajectories via a *cost function*. The cost function uses a set of *penalties* for penalizing trajectories that require driving characteristics over a critical limit (e.g., speed limit, lateral acceleration over a given threshold). These penalties are configurable (i.e., they are *configuration options* of `MP`). In the same scenario, `MP` can behave differently under different penalty configurations, and so this may result in different requirements violations.

For example, let us consider two similar scenarios, i.e., *Scenario-1* and *Scenario-2* shown in Fig. 2: the *ego-vehicle* plans to turn right from *lane-1* into *lane-2* at the intersection, while *vehicle-a* is crossing the intersection from bottom to up

and *vehicle-b* is crossing from right to left. The difference between the two scenarios is that in *Scenario-1*, *ego-vehicle* proceeds slower at the start. Let us consider two configurations for the penalties of the MP cost function, i.e., *Configuration-A* and *Configuration-B*. MP under *Configuration-B* is more aggressive and cares less of exceeding the threshold of lateral acceleration than under *Configuration-A*.

In *Scenario-1* (Figs. 2(a) and 2(b)), MP under both *Configuration-A* and *Configuration-B* orders an emergency braking to bring the *ego vehicle* to a halt; however, the *ego vehicle* cannot stop in time and collides with *vehicle-a* (i.e., violating R_2). Although the *ego vehicle* collides under both configurations, the collision under *Configuration-B* is more serious, considering the higher relative collision speed (17 m/s) than that under *Configuration-A* (15 m/s).

In *Scenario-2* in which *ego-vehicle* proceeds slightly faster than in *Scenario-1*, the *ego vehicle* under *Configuration-A* also collides (i.e., violating R_2), as shown in Fig. 2(c). Instead, the *ego vehicle* under *Configuration-B* accelerates and tries to turn before *vehicle-a* crossed the intersection. This maneuver requires a higher lateral acceleration $11.4m/s^2$ for the *ego vehicle*, to the point to exceed the limit $6.86m/s^2$ (i.e., violating R_7), as shown in Fig. 2(d). Although the *ego vehicle* under the two configurations violates some requirements, the behavior under *Configuration-A* is more dangerous, as the violation of R_2 is more serious than that of R_7. From these examples, we can see that devising a safe configuration requires a systematic assessment of the severity of safety requirements violations.

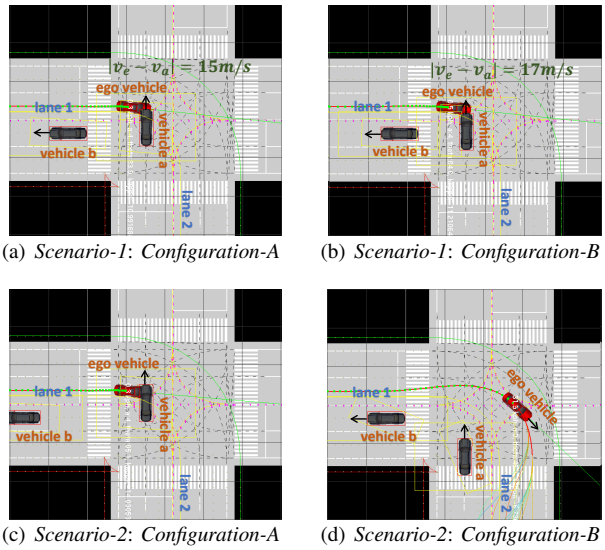
Assessing the safety for behaviors of MP under different configurations, requires a quantitative model to analyze requirements violations. Furthermore, under a certain scenario, systematically comparing the severity of violations of safety requirements can be particularly challenging, as both the severity of violated requirements and the importance of the requirements matter. We need approaches that can (1) quantitatively analyze the violations of safety requirements for MP under different configurations in given scenarios; (2) comprehensively compare the analysis results of requirements violations by giving scenarios in specific traffic situations.

III. Formalization

We here provide the definition of configurable ADS and model its safety requirements.

A. Configurable Autonomous Driving System

According to existing works [14], [15], an ADS is a highly configurable cyber-physical system whose behavioral decisions depend on the state of the ego vehicle. The kinematic states of the ego vehicle can be described as a tuple of three elements, i.e., $\boldsymbol{s}_k^e = (\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{a}_k)$ at time instant k. Vector $\boldsymbol{p}_k = \langle x, y \rangle$ is the geometric center of the position of the ego vehicle, while $\boldsymbol{v}_k$ and $\boldsymbol{a}_k$ are its velocity and acceleration. The trajectory of the ego vehicle is a sequence of vehicle's states, i.e., $\mathcal{T} = [\boldsymbol{s}_0^e, \ldots, \boldsymbol{s}_T^e]$, where the time interval between two consecutive states is fixed as a parameter of the simulator. T is the simulation time duration.



(a) *Scenario-1*: *Configuration-A* (b) *Scenario-1*: *Configuration-B*

(c) *Scenario-2*: *Configuration-A* (d) *Scenario-2*: *Configuration-B*

Fig. 2: Behaviors of the ego vehicle under different configurations in given scenarios (left-hand traffic)

The ego vehicle continuously interacts with dynamic and uncertain environments. A scenario t describes the environment in which the ego vehicle is operating, including: (i) a map of the road structure; (ii) the traffic regulations enforced by law within a geographical area (e.g., speed limit); (iii) initial state of the ego vehicle $\boldsymbol{s}_0^e$; (iv) destination for the ego vehicle $\boldsymbol{p}_d$; (v) dynamic behaviors of the external objects ($[\boldsymbol{s}_0^o, \ldots, \boldsymbol{s}_T^o]$, $o \in \mathcal{O}$)[1]; (vi) duration of the simulation T.

The ADS can have different configurations. Let C_i be the i-th configuration option, which ranges over a finite domain Dom_i. The configuration space is the Cartesian product of the domains of the parameters of interest $\mathbb{C} = Dom_1 \times \cdots \times Dom_m$, where m is the total number of configuration options. A configuration $\boldsymbol{c}$ then is a vector in the configuration space $\boldsymbol{c} \in \mathbb{C}$ that assigns a particular value along each dimension. We denote as $ADS_{\boldsymbol{c}}$ the ADS configured with $\boldsymbol{c} = \langle c_1, \ldots, c_m \rangle$, $c_i \in Dom_i$. The ADS under a certain configuration $\boldsymbol{c}$ can be seen as a function that, given scenario t, produces the trajectory of the ego vehicle, i.e., $\mathcal{T} = ADS_{\boldsymbol{c}}(t)$.

B. Safety Requirements Modeling

As the ego vehicle is a part of the traffic, the ADS has to achieve safety requirements from both the perspectives of the system itself and the environment [2]. The environment-centric safety requirements can be elicited from different environment entities that the ego vehicle interacts with, e.g., passengers, other vehicles, and constraints like traffic regulations.

We use the goal model notation to model the safety requirements, i.e., decompose the high-level safety requirements into several subgoals, as done in existing works for autonomous systems [16], [17]. In this way, a set of atomic leaf goals

[1] $\mathcal{O}$ is a set of external objects that interact with the ego vehicle, e.g., pedestrians, other vehicles; their kinematic states are modeled in the same way as the ego vehicle.

can be derived, which is denoted as $\mathcal{R} = \{R_1, \ldots, R_n\}$ in this paper. To quantitatively evaluate the violation of leaf safety requirements, the Safety Metric $\mathcal{X}_i$ is introduced. $\mathcal{X}_i$ is defined as a function of the trajectory $\mathcal{T}$ of the ego vehicle and the running scenario t, i.e., $\mathcal{X}_i = h_i(\mathcal{T}, t)$. Its value at time instant k is denoted as $\mathcal{X}_i(k)$. When comparing the value $\mathcal{X}_i(k)$ with the specified target value g_i (i.e., $\mathcal{X}_i(k) \bowtie g_i$), there are three types of expected relationship, i.e., $\bowtie \in \{\leq, \geq, \approx\}$. In case that the expected relationship is deviated, requirement R_i is violated at time instant k. All the Safety Metrics of the requirements described in § II are reported in the supplementary materials online [18].

C. Requirements Prioritization

For the ADS operating in a complex environment, satisfying all the requirements may not always be possible, so it is important to identify the important ones under certain situations. In the existing work of requirements prioritization, different criteria have been considered: value/benefit, cost, risk, penalty, etc. [19]. Here, we prioritize the safety requirements according to the *criticality* of the violations. Given the initial set of atomic safety requirements $\mathcal{R} = \{R_1, \ldots, R_n\}$, it can be partitioned into several subsets by requirements importance, i.e., $\mathcal{R} = \mathcal{R}_{\mathtt{I}} \cup \mathcal{R}_{\mathtt{II}} \cup \ldots \cup \mathcal{R}_{\mathtt{N}}$ and $\mathcal{R}_{\mathtt{P}} \cap \mathcal{R}_{\mathtt{Q}} = \emptyset$ for $\mathtt{I} \leq \mathtt{P} < \mathtt{Q} \leq \mathtt{N}$. The partition is obtained with this rationale: the satisfaction of any requirement R_i in $\mathcal{R}_{\mathtt{P}}$ is more important than the simultaneous satisfaction of all the requirements belonging to all $\mathcal{R}_{\mathtt{Q}}$, for each $\mathtt{Q} > \mathtt{P}$. We say that the importance level for a certain partition of requirements $\mathcal{R}_{\mathtt{P}}$ is $\mathtt{P}$. The higher the importance level is (i.e., lower index), the more serious the adverse impacts are if requirements belonging to this partition are violated. ADS engineers can assign different importance levels to each requirement in different traffic situations, so the partition may change along with traffic situations.

Example 1. As described in § II, there are seven requirements considered by MP, i.e., $\mathcal{R} = \{R_1, \ldots, R_7\}$. We partition these requirements into four importance levels I to IV, i.e., $\mathcal{R} = \mathcal{R}_{\mathtt{I}} \cup \mathcal{R}_{\mathtt{II}} \cup \mathcal{R}_{\mathtt{III}} \cup \mathcal{R}_{\mathtt{IV}}$, following the suggestions from our industry partner and a guideline [3] in safety requirements modeling for ADS:

- $\mathcal{R}_{\mathtt{I}} = \{R_1\}$ (**Vehicle Stability**): assure the stable control and avoid impossible actions of the vehicle.
- $\mathcal{R}_{\mathtt{II}} = \{R_2\}$ (**Safe Distance**): keep a safe distance with other objects along the trajectory.
- $\mathcal{R}_{\mathtt{III}} = \{R_3, R_4\}$ (**Compliance**): respect the traffic regulations enforced by law in a geographical area.
- $\mathcal{R}_{\mathtt{IV}} = \{R_5, R_6, R_7\}$ (**Smoothness**): plan a smooth trajectory to prevent passengers from being hurt.

We assume that the requirement that affects the stability of the vehicle belongs to the highest importance level I, as losing control of the ego vehicle may result in colliding with other road users or objects, skidding off the roadway, and rollover. The second importance level contains the requirement related to a safe distance, which requires the car to keep the minimum distance to the other cars and provide enough time to brake to avoid a collision. The speed limit and lane-keeping requirements, which are formal traffic regulations, rank third. Although driving fast can lead to accidents, it is also needed to avoid a collision in some cases. Similarly, it is sometimes necessary to invade the other lane to avoid a collision. The fourth partition contains three requirements indicating good driving practices that allow smooth travel for passengers.

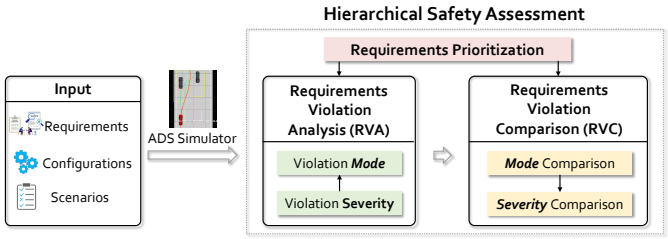


Fig. 3: Overview of the hierarchical safety assessment (HSA)

D. Problem Statement

For the ADS under different configurations, an assessment A of safety requirements $\mathcal{R}$ under a set of scenarios $\mathcal{TS}$, can be defined as a function $\overrightarrow{\mathcal{C}} = A(ADS, \mathcal{C}, \mathcal{R}, \mathcal{TS})$. $\mathcal{C} \subseteq \mathbb{C}$ is a set of alternative configurations to be assessed and $\mathcal{R}$ is a set of safety requirements for the ADS to achieve. $\overrightarrow{\mathcal{C}}$ is the output of A, i.e., a ranking list of these configurations, from the perspective of safety requirements violations.

IV. Hierarchical Safety Assessment

A. Overview

To differentiate the safety of the differently configured ADS, it is necessary to summarize the differences in safety requirements violations in a quantifiable and systematic way before drawing any conclusion on which configuration option is safer. Fig. 3 shows the workflow of HSA for a configurable ADS. HSA receives a set of safety requirements $\mathcal{R}$, a set of ADS configurations $\mathcal{C}$ and a set of scenarios $\mathcal{TS}$ as inputs. After simulating the ADS and recording trace logs in each scenario of $\mathcal{TS}$, RVA is used to quantitatively evaluate the violations of requirements for each configuration in each scenario. Then, RVC is used to hierarchically compare the RVA results by requirements importance level for different configurations in pairs. The combination of RVA and RVC enables HSA to assess the behaviors of ADS under different configurations and so rank these in terms of safety.

More specifically, HSA works as follows. In the beginning, in RVA (see § IV-B), given the ADS safety requirements, the different ADS configurations $\mathcal{C}$, and a set of scenarios $\mathcal{TS}$, we compute the Safety Metric $\mathcal{X}$ of the behaviors of ADS under configurations $\boldsymbol{c} \in \mathcal{C}$ in each scenario $t \in \mathcal{TS}$ by running the ADS simulator. By comparing $\mathcal{X}$ and the target value g, we analyze two questions: (i) how severe the violations of requirements are, and (ii) which importance level the violated requirements belong to. The answers to these questions are the results of RVA and are denoted as *requirements violation severity* and *requirements violation mode*, respectively.

After the analysis of RVA, the sorting of the selected configurations starts by employing RVC in pairs (see § IV-C). Concretely, when comparing two configurations, RVC works as follows. It incrementally checks whether one configuration can beat the other one over the set of scenarios, by considering the requirements in the order defined by the importance levels, from I to N. When two configurations tie in the previous importance level, it goes down to the next importance level. RVC is terminated when it can distinguish between the two configurations or reaches the last level. The application of RVC to all the pairs of configurations leads to the definition of a ranked list of configurations in terms of safety.

B. Requirements Violation Analysis (RVA)

To evaluate the violation of safety requirements from the behaviors of the ADS during its dynamic interaction with operating environments, previous works [14], [20], [21] propose to use a mapping function to indicate whether a requirement is satisfied/violated. However, these works only consider whether a requirement is violated or not at some point during the ADS execution, but do not consider to *what degree* the requirement is violated in a specific time moment, *how long* a violation lasts, and *how many times* the violation happens.

To fill this gap, we evaluate the violation severity for each requirement from three aspects: the *violation degree*, the *duration of the violation*, and the *number of violations*. Given a set of safety requirements $\mathcal{R}$ along with the Safety Metrics $\mathcal{X}$ as defined in § III-B, the *violation degree* for requirement R_i can be computed based on the deviation of $\mathcal{X}_i(k)$ ($k \in [0, T]$) from target value g_i at time instant k. For the expected LESS THAN ($\leq$) relationship between $\mathcal{X}_i(k)$ and g_i, the violation degree for the requirement at time instant k can be calculated as $D_k^{R_i} = \frac{1}{|g_i|} \cdot \max(\mathcal{X}_i(k) - g_i, 0)$. For the expected MORE THAN ($\geq$) relationship, the violation degree is calculated as $D_k^{R_i} = \frac{1}{|g_i|} \cdot \max(g_i - \mathcal{X}_i(k), 0)$. For the expected AS CLOSE AS POSSIBLE ($\approx$) relationship, the violation degree is calculated as $D_k^{R_i} = \frac{1}{\epsilon} \cdot \max\big(\mathcal{X}_i(k) - (g_i + \epsilon), (g_i - \epsilon) - \mathcal{X}_i(k), 0\big)$, where ϵ is an acceptable deviation from the target value.

Example 2. Requirement R_3 states that the velocity of the ego vehicle should be always less than the speed limit in a geographical area. The Safety Metric of this requirement is the speed of the ego vehicle along the trajectory $\mathcal{X}_3(k) = \boldsymbol{v}_k^e$, and $\mathcal{X}_3(k)$ should be LESS THAN ($\leq$) the speed limit as v_{limit}. The violation degree can be calculated as $D_k^{R_3} = \frac{1}{v_{limit}} \max(\mathcal{X}_3(k) - v_{limit}, 0)$.

To calculate the duration of each requirement violation and the number of violations, we introduce a Boolean violation indicator $y_k^{R_i}$ describing whether R_i is violated at time instant k. $y_k^{R_i}$ is false if the violation degree $D_k^{R_i}$ is zero, otherwise $y_k^{R_i}$ is true. Then, the violation sequence of the requirement R_i along the trajectory is defined as $vs^{R_i} = [y_0^{R_i}, \ldots, y_T^{R_i}]$.

Definition 1 (Requirements Violation Duration)**.** Given a requirements violation sequence vs^{R_i}, we identify the *violation duration*(s) of R_i as the consecutive time instants in which the requirement is violated, i.e., $VD(vs^{R_i}) = \{[\alpha_1, \beta_1], \ldots, [\alpha_d, \beta_d]\}$[2] such that:

- $\alpha_h \leq \beta_h$ (for $h = 1, \ldots, d$), $\beta_h < \alpha_{h+1}$ (for $h = 1, \ldots, d-1$), $\alpha_1 \geq 0$, $\beta_d \leq T$;
- $\forall k \in [\alpha_h, \beta_h] : y_k^{R_i}$ (for $h = 1, \ldots, d$);
- $\forall k \in [\beta_h + 1, \alpha_{h+1} - 1] : \neg y_k^{R_i}$ (for $h = 1, \ldots, d-1$);
- $\forall k \in [0, \max(0, \alpha_1 - 1)] : \neg y_k^{R_i}$, $\forall k \in [\beta_d + 1, \min(\beta_d + 1, T)] : \neg y_k^{R_i}$.

Based on Def. 1, the duration of each violation is the length of consecutive time instants, i.e., $\alpha_h - \beta_h$, while the number of violations is d. Therefore, the violation severity S_i for requirement R_i is calculated by the estimation of accumulated impacts of violations as defined in Eq. (1). The higher the violation degree is, the more times the violation happens, and the longer it lasts, all lead to more severe consequences, e.g., rolling over or colliding with other vehicles.

$$S_i = \sum_{h=1}^{d} \sum_{k=\alpha_h}^{\beta_h} D_k^{R_i} \cdot \exp(k - \alpha_h) \tag{1}$$

Definition 2 (Requirements Violation Severity)**.** Let S_i indicate the violation severity for requirement R_i, whose domain is D_i. The space of the violation analysis results for all requirements is the Cartesian product of the domains $\mathbb{S} = D_1 \times \cdots \times D_n$. The *Requirements Violation Severity* is a vector $\boldsymbol{S}_{\boldsymbol{c}}^t = [S_1, \ldots, S_n] \in \mathbb{S}$, representing the violation severity of $\mathcal{R}$ for the behaviors of the ADS under configuration $\boldsymbol{c}$ when running scenario t.

Based on Def. 2, if $S_i > 0$, we can say requirement R_i is violated in this scenario, otherwise (i.e., $S_i = 0$), R_i is satisfied. Note that the Safety Metrics may have different units (e.g., km/h, meters) and different ranges. For fair comparison of the violation severity between different requirements, we normalize the violation severity into the range $[0, 1)$, relying on the well-known rational function $\omega(x) = x/(x+1)$ [22]. In the following, we denote the normalized requirements violation severity as $\overline{\boldsymbol{S}_{\boldsymbol{c}}^t} = [\overline{S_1}, \ldots, \overline{S_n}]$.

Definition 3 (Requirements Violation Mode)**.** Given the partition of requirements by importance (see § III-C), i.e., $\boldsymbol{R} = \mathcal{R}_{\mathtt{I}} \cup \ldots \cup \mathcal{R}_{\mathtt{N}}$, and normalized requirements violation severity $\overline{\boldsymbol{S}_{\boldsymbol{c}}^t}$ for $ADS_{\boldsymbol{c}}$ in scenario t, the *Requirements Violation Mode* $\boldsymbol{M}_{\boldsymbol{c}}^t = [M_{\mathtt{I}}, M_{\mathtt{II}}, \ldots, M_{\mathtt{N}}]$ is a vector that represents the number of violated requirements by each importance level. $M_{\mathtt{K}} \in \{0, \ldots, |\mathcal{R}_{\mathtt{K}}|\}$ indicates the number of violated requirements belonging to the importance level $\mathcal{R}_{\mathtt{K}}$.

Example 3. For MP, $\overline{\boldsymbol{S}_{\boldsymbol{c}}^t} = [0, 0, 0, 0, 0, 0.6, 0.8]$ represents the normalized requirements violation severity with the evaluation of seven requirements, where the last two are violated, and the first five are satisfied. Based on the partition of seven requirements by importance, the corresponding requirements violation mode is $\boldsymbol{M}_{\boldsymbol{c}}^t = [0, 0, 0, 2]$. Given the importance

[2] $[a, b]$ identifies an integer interval and $b - a$ is its length.

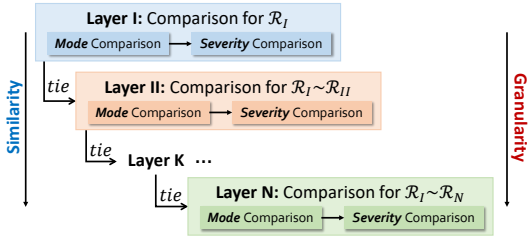


Fig. 4: Layout of Requirements Violation Comparison (RVC)

levels of the seven requirements (see § II), there are $2 \times 2 \times 3 \times 4 = 48$ modes altogether.

C. Requirements Violation Comparison (RVC)

We here describe the details of the Requirements Violation Comparison (RVC), i.e., the second step of the approach that assesses the safety of different configurations over the set of scenarios based on the requirements prioritization by importance (see § III-B) and requirements violation analysis (RVA, § IV-B).

The layout of RVC is shown in Fig. 4. To assess the safety of two configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$, it considers the analysis results of RVA starting from the first requirements importance level I to the last one N. The idea is that the violation of requirements with higher importance levels is more severe than that with lower importance levels. Thus, if $ADS_{\boldsymbol{c}}$ exposes less violations in higher importance levels than $ADS_{\boldsymbol{c}'}$, we say that $\boldsymbol{c}$ is safer. Based on this heuristic rule, the algorithm works in a hierarchical way as follows. In the first layer, it compares the violation modes and severity of requirements belonging to the importance level I, i.e., $R_i \in \mathcal{R}_{\mathtt{I}}$. When the RVA results for ADS under configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ tie in layer I, the violation severity of requirements belonging to $\mathcal{R}_{\mathtt{I}}$ and $\mathcal{R}_{\mathtt{II}}$ are compared in the second layer. This process continues until $\boldsymbol{c}$ and $\boldsymbol{c}'$ can be distinguished, or it reaches the final layer N in which the violation severity of all requirements are compared. We can observe that from layer I to layer N, the comparison granularity becomes finer and the similarity of RVA results between $ADS_{\boldsymbol{c}}$ and $ADS_{\boldsymbol{c}'}$ increases.

Alg. 1 shows the implementation of RVC. It receives the evaluation results of RVA for the ADS configured with $\boldsymbol{c}$ and $\boldsymbol{c}'$ over the complete set of scenarios $\mathcal{TS}$, including two sets of requirements violation modes and requirements violation severity, i.e., $\mathcal{M}_{\boldsymbol{c}} = \{\boldsymbol{M}^t_{\boldsymbol{c}}\}, \mathcal{M}_{\boldsymbol{c}'} = \{\boldsymbol{M}^t_{\boldsymbol{c}'}\}, \mathcal{S}_{\boldsymbol{c}} = \{\boldsymbol{S}^t_{\boldsymbol{c}}\}, \mathcal{S}_{\boldsymbol{c}'} = \{\boldsymbol{S}^t_{\boldsymbol{c}'}\}, t \in \mathcal{TS}$. The output of RVC is the comparison result between configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ in safety assessment. RVC incrementally performs from layer I to layer N (Line 2), until two configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ can be told apart. The comparison result $RVCres$ describes whether $\boldsymbol{c}$ ties with $\boldsymbol{c}'$ (Line 16), or $\boldsymbol{c}$ and $\boldsymbol{c}'$ are distinguished (Lines 11 and 14); in the latter case, it reports at which layer, by which requirements violation mode, and by which safety requirement they are distinguished.

Specifically, the algorithm works as follows. For the comparison in the K-th layer, it compares the RVA results of requirements in $\mathcal{R}_{\mathtt{I}} \cup \cdots \cup \mathcal{R}_{\mathtt{K}}$. It first extracts the first K

Algorithm 1: Requirements Violation Comparison

Input: $\mathcal{M}_{\boldsymbol{c}}, \mathcal{M}_{\boldsymbol{c}'}$: two sets of requirements violation mode;
$\mathcal{S}_{\boldsymbol{c}}, \mathcal{S}_{\boldsymbol{c}'}$: two sets of requirements violation severity;
N: total number of requirements importance levels.
Output: $RVCres$: comparison results of $\boldsymbol{c}$ and $\boldsymbol{c}'$

1 $RVCres \leftarrow (\boldsymbol{c}' = \boldsymbol{c})$;
2 **for** $K = I \ldots N$ **do**
3 $\quad A \leftarrow \{\boldsymbol{M}^t_{\boldsymbol{c}}[\mathtt{I}:\mathtt{K}] \mid \boldsymbol{M}^t_{\boldsymbol{c}} \in \mathcal{M}_{\boldsymbol{c}}\}$;
4 $\quad B \leftarrow \{\boldsymbol{M}^t_{\boldsymbol{c}'}[\mathtt{I}:\mathtt{K}] \mid \boldsymbol{M}^t_{\boldsymbol{c}'} \in \mathcal{M}_{\boldsymbol{c}'}\}$;
5 $\quad \mathcal{M}_s \leftarrow prioritize(A \cup B)$;
6 $\quad$ **for** $i = 1, \ldots, |\mathcal{M}_s|$ **do**
7 $\quad\quad SS_{\boldsymbol{c}} \leftarrow \sum_t \boldsymbol{S}^t_{\boldsymbol{c}}, t \in \{x \in \mathcal{TS} \mid \boldsymbol{M}^t_{\boldsymbol{c}}[\mathtt{I}:\mathtt{K}] = \mathcal{M}_s[i]\}$;
8 $\quad\quad SS_{\boldsymbol{c}'} \leftarrow \sum_t \boldsymbol{S}^t_{\boldsymbol{c}'}, t \in \{t \in \mathcal{TS} \mid \boldsymbol{M}^t_{\boldsymbol{c}'}[\mathtt{I}:\mathtt{K}] = \mathcal{M}_s[i]\}$;
9 $\quad\quad$ **for** $j = I, \ldots, K$ **do**
10 $\quad\quad\quad$ **if** $SS_{\boldsymbol{c}}[j] > SS_{\boldsymbol{c}'}[j]$ **then**
11 $\quad\quad\quad\quad RVCres \leftarrow (\mathtt{K}, \mathcal{M}_s[i], j, \boldsymbol{c}' \succ \boldsymbol{c})$;
12 $\quad\quad\quad\quad$ **Return** comparison results $RVCres$;
13 $\quad\quad\quad$ **else if** $SS_{\boldsymbol{c}}[j] < SS_{\boldsymbol{c}'}[j]$ **then**
14 $\quad\quad\quad\quad RVCres \leftarrow (\mathtt{K}, \mathcal{M}_s[i], j, \boldsymbol{c} \succ \boldsymbol{c}')$;
15 $\quad\quad\quad\quad$ **Return** comparison results $RVCres$;
16 **Return** comparison results $RVCres$

elements of requirements violation modes (i.e., $\boldsymbol{M}^t_{\boldsymbol{c}}[\mathtt{I}:\mathtt{K}]$ and $\boldsymbol{M}^t_{\boldsymbol{c}'}[\mathtt{I}:\mathtt{K}]$) for configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ as sets A and B, respectively (Lines 3-4), i.e., it limits them to modes from $\mathcal{R}_{\mathtt{I}}$ to $\mathcal{R}_{\mathtt{K}}$. Then, based on Def. 3 and the rules of requirements prioritization by importance in § III-B, it prioritizes the elements $\boldsymbol{M} \in A \cup B$ following the ranking rule $prioritize$, obtaining, as output, the sorted list $\mathcal{M}_s$ (Line 5). In $\mathcal{M}_s$, $\boldsymbol{M}'$ ranks higher than $\boldsymbol{M}$ (referred to as $\boldsymbol{M}' \succ \boldsymbol{M}$) if: (1) $\boldsymbol{M}'$ has more violated requirements than $\boldsymbol{M}$ in a given importance level L (i.e., $\exists \mathtt{P} \in \{\mathtt{I}, \ldots, \mathtt{K}\}: \boldsymbol{M}'[\mathtt{P}] > \boldsymbol{M}[\mathtt{P}]$), and (2) $\boldsymbol{M}$ and $\boldsymbol{M}'$ have the same number of violated requirements in importance levels preceding P (i.e., $\forall \mathtt{Q} \in \{\mathtt{I}, \ldots, \mathtt{P}-1\}: \boldsymbol{M}[\mathtt{Q}] = \boldsymbol{M}'[\mathtt{Q}]$). For example, $(\boldsymbol{M}' = [0,1,1,0]) \succ (\boldsymbol{M} = [0,0,1,1])$, as $1 > 0$ in importance level II.

Then, for each $\mathcal{M}_s[i] \in \mathcal{M}_s$ (Line 6), it sums up the violation severity for the scenarios whose first K bits of their violation modes are the same with $\mathcal{M}_s[i]$ for configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ as $SS_{\boldsymbol{c}}$ and $SS_{\boldsymbol{c}'}$, respectively (Lines 7 and 8). Therefore, the total severity $SS_{\boldsymbol{c}}$ is a vector sum of K dimensions where the j-th dimension, i.e., $SS_{\boldsymbol{c}}[j]$, indicates the overall violation severity for requirements belonging to importance level j. The value of $SS_{\boldsymbol{c}}$ is affected by two aspects: (i) the number of scenarios whose violation modes are consistent with $\mathcal{M}_s[i]$; (ii) the value of violation severity in these scenarios.

Finally, the algorithm compares vectors $SS_{\boldsymbol{c}}$ and $SS_{\boldsymbol{c}'}$ starting from I to K (Line 9), until either $SS_{\boldsymbol{c}}[j] > SS_{\boldsymbol{c}'}[j]$ or $SS_{\boldsymbol{c}}[j] < SS_{\boldsymbol{c}'}[j]$ occurs: in this case, we can draw the conclusion that $\boldsymbol{c}'$ or $\boldsymbol{c}$ is safer, respectively. Instead, if no difference is observed, the algorithm returns a result saying that $\boldsymbol{c}'$ and $\boldsymbol{c}$ are equivalent.

RVC defines a binary relation between the configurations in $\mathcal{C}$ that leads to a total order. Proof of RVC's transitivity (i.e., if $\boldsymbol{c}_1 \succeq \boldsymbol{c}_2$ and $\boldsymbol{c}_2 \succeq \boldsymbol{c}_3$, then $\boldsymbol{c}_1 \succeq \boldsymbol{c}_3$) is available in the supplementary material [18]. Therefore, the application of RVC among all the pairs of configurations in $\mathcal{C}$ is done using a classical sorting algorithm that returns the total order $\overrightarrow{\mathcal{C}}$.

TABLE I: Six traffic situations used in the experiments

Name	Description
$t_{\mathcal{SA}}$	The ego vehicle is proceeding on its lane, while *vehicle-a* and *vehicle-b* Aside are crossing from the Same direction (right to left).
$t_{\mathcal{M}}$	The ego vehicle is proceeding on its lane when it meets *vehicle-a* proceeding in the opposite direction in the other lane.
$t_{\mathcal{H}}$	The ego vehicle must turn right at the intersection, and there is *vehicle-a* crossing from the opposite direction which is Hidden by a queue of vehicles waiting at the intersection, and *vehicle-b* crossing from left.
$t_{\mathcal{DA}}$	The ego vehicle is proceeding on its lane and *vehicle-a* and *vehicle-b* Aside are crossing from Different directions, i.e., *vehicle-a* from left and *vehicle-b* from right.
$t_{\mathcal{O}}$	The ego vehicle tries to Overtake *vehicle-a* proceeding slowly, while *vehicle-b* is coming from behind on the passing lane.
$t_{\mathcal{T}}$	The ego vehicle must Turn right at the intersection. *vehicle-a* is crossing from the opposite direction and *vehicle-b* is crossing from right.

Hence, its complexity is $n\log(n)$, with $n = |\mathcal{C}|$.

V. Evaluation

We here present the application of HSA to the case study system MP provided by our industry partner; the safety requirements (see § II) are elicited with the help of its engineers.

In order to demonstrate the effectiveness of HSA, we investigate the following research questions.

RQ1: *How is the feasibility of HSA in distinguishing configurations with respect to safety requirements violations?*

RQ2: *Are the configurations identified as safer by HSA applicable to most scenarios in a given traffic situation?*

We also validate whether HSA is useful in ADS configuring:

RQ3: *Can HSA provide useful information for configuring a safe ADS in an industrial setting?*

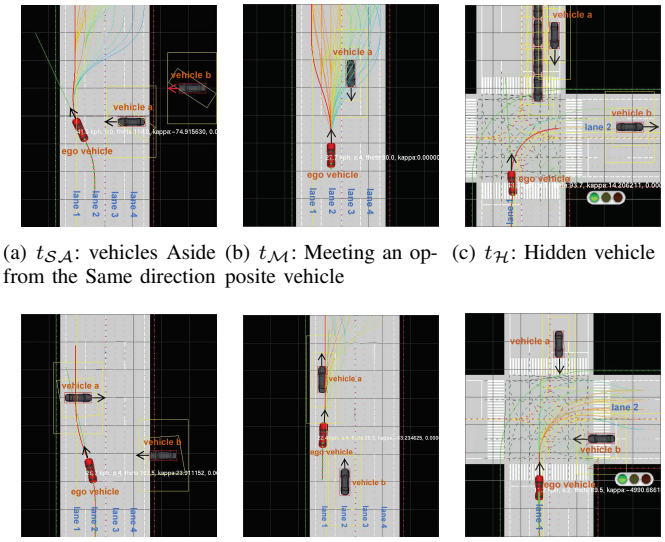
A. Experimental Design and Settings

1) Traffic Scenarios: To evaluate HSA under different conditions, we identify six *traffic situations*, as shown in Table I (assuming left-hand traffic). Five traffic situations (i.e., $t_{\mathcal{SA}}$, $t_{\mathcal{M}}$, $t_{\mathcal{DA}}$, $t_{\mathcal{O}}$, $t_{\mathcal{T}}$) have been designed by our industry partner, while $t_{\mathcal{H}}$ reproduces the scenario of a real collision of a self-driving Uber car, which was also considered in [23].

A traffic situation (also called *logical scenario* in ADS testing [24]) characterizes a set of similar *concrete scenarios*; namely, it specifies a *baseline scenario* defining some characteristics shared by all the concrete scenarios (e.g., road map, speed limit), and specific characteristics (e.g., position, speed, and acceleration of the ego vehicle and other vehicles) which are left as variables ranging over a defined space. Fig. 5 reports snapshots of the six traffic situations. The defined space of their variables can be found online [18].

In order to apply HSA in a given traffic situation, we need to select some concrete scenarios from it. In this work, we obtained these scenarios by the algorithm of random generation [25]. Totally, we fix the number of generated scenarios as 10000 in each traffic situation, and the time budget T for each scenario simulation is set as 100 seconds.

2) ADS Configurations: The goal of this work is to assess how different configurations of an ADS affect the satisfaction of safety requirements. Since we cannot evaluate all the possible configurations, we need to select some of them. For the case study system MP, we modify its six configuration



(a) $t_{\mathcal{SA}}$: vehicles Aside from the Same direction (b) $t_{\mathcal{M}}$: Meeting an opposite vehicle (c) $t_{\mathcal{H}}$: Hidden vehicle

(d) $t_{\mathcal{DA}}$: vehicles Aside from Different directions (e) $t_{\mathcal{O}}$: Overtake a vehicle proceeding slowly (f) $t_{\mathcal{T}}$: Turn right

Fig. 5: Snapshots of traffic situations used in the experiments

options C_{cur}, C_{dis}, C_{spd}, C_{acc}, C_{dec}, C_{latg}, starting from its original configuration $\boldsymbol{c}^0 = \langle c^0_{cur}, \ldots, c^0_{latg}\rangle$. Each modified configuration $\boldsymbol{c}'$ differs from $\boldsymbol{c}^0$ only in the value of a configuration option C_i, which is obtained by multiplying the original value c^0_i by a constant γ, i.e., $c'_i = \gamma \cdot c^0_i$. In order to explore different ranges for each configuration option, we use the following ten values: $\gamma = \{\frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 2, 4, 8, 16, 32\}$. In total, we have $6 \times 10 = 60$ modified configurations of MP. γ is chosen to sample the space of possible configuration values. In particular, $\gamma[1] = \frac{1}{32}$ and $\gamma[10] = 32$ show extreme changes. The other values of γ let us explore the effect of different scales of change on the configuration values.

3) Compared approach: To the best of our knowledge, there is no approach that assesses the different configurations of an ADS with respect to multiple safety requirements. In the absence of such an approach, one would consider an ADS configuration better than another one only if it is better for all the requirements in all the scenarios; we call such an approach as Conservative Comparison (CC). It can be seen as a strict safety assessment based on the analysis results of RVA, in which configuration $\boldsymbol{c}$ beats another configuration $\boldsymbol{c}'$ only if $(\forall i \in \{1,\ldots,n\}, t \in \mathcal{TS}\colon \boldsymbol{S}^t_{\boldsymbol{c}}[i] \leq \boldsymbol{S}^t_{\boldsymbol{c}'}[i]) \wedge (\exists i \in \{1,\ldots,n\}, t \in \mathcal{TS}\colon \boldsymbol{S}^t_{\boldsymbol{c}}[i] < \boldsymbol{S}^t_{\boldsymbol{c}'}[i])$. We compare HSA to it to check whether CC is a feasible approach and the approximation done by HSA is necessary.

We implemented CC and HSA in Python. Experiments were executed on servers with the CPU (Intel Xeon E5-2697A V4@2.6GHz), 32 cores, and 256 GB of RAM. Complete experimental results and the implementation of the approach can be found online [18].

B. Results and Analysis

1) Evaluation of RQ1: We investigate the ability of CC and of HSA to distinguish ADS configurations, i.e., to what extent

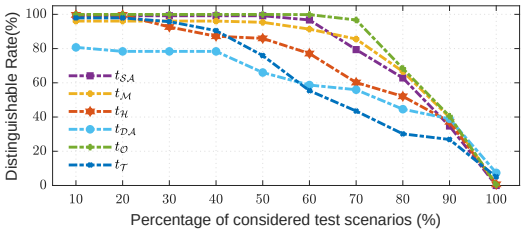


Fig. 6: RQ1–Distinguishable rate of Conservative Comparison

TABLE II: RQ1–Percentage of comparison distinguished in each layer in RVC (all scenarios in $\mathcal{TS}$ are considered)

Layers	$t_{\mathcal{SA}}$	$t_{\mathcal{M}}$	$t_{\mathcal{H}}$	$t_{\mathcal{DA}}$	$t_{\mathcal{O}}$	$t_{\mathcal{T}}$
I	87.98%	63.28%	85.79%	3.28%	92.02%	81.64%
II	6.12%	33.17%	8.42%	72.79%	6.72%	6.67%
III	2.02%	0.22%	5.25%	-	1.20%	-
IV	3.06%	-	0.22%	17.60%	-	9.78%
Total	99.18%	96.67%	99.67%	93.66%	99.95%	98.09%

they are able to differentiate two configurations under the analysis results of RVA. Since there is an original configuration $\boldsymbol{c}^0$ and 60 modified configurations $\boldsymbol{c}'_i$ (see § V-A2), there are altogether $C^2_{61} = 1830$ pairs of configurations to compare.[3]

In Fig. 6, we report the *distinguishable rate* (i.e., the percentage of pairs of configurations that can be distinguished) of CC applied to an increasing number of scenarios (i.e., the percentage of the complete set of scenarios $\mathcal{TS}$). We observe that the distinguishable rate decreases as the percentage of considered scenarios increases from 10% to 100%, and ends up with less than 10%. The highest distinguishable rate when considering all the scenarios is 7.38% in $t_{\mathcal{DA}}$ and the lowest one is 0% in $t_{\mathcal{O}}$. Indeed, as the number of considered scenarios increases, the criterion used by CC to distinguish becomes more strict, so it is harder to distinguish configurations. This shows that CC is not a feasible approach to distinguish configurations when applied to a large number of scenarios.

By construction, HSA is able to distinguish more configurations, as it does not apply a strict criterion in the comparison, but checks which configuration is "overall" better in a given layer. We report in Table II the percentage of pairs of configurations that are distinguished in each layer, when all the scenarios in $\mathcal{TS}$ are considered. The distinguishable rates of our approach in the six traffic situations are all over 93%; these are all the cases in which RVC can identify a better configuration (Line 12 or Line 15 in Alg. 1). In the remaining cases, RVC does not assess any difference between the compared configurations (Line 16 in Alg. 1).

We observe that in most traffic situations, as the layer increases from I to IV, the percentage of configurations pairs distinguished by that layer decreases. Indeed, as shown in the layout of RVC in Fig. 4, as the layer increases, the similarity of RVA results of two configurations increases as well. Thus, it is reasonable that the approach distinguishes more in the first layer in which it is more likely that the RVA results are different. In some special cases (i.e., layers I and III in $t_{\mathcal{DA}}$ and layer III in $t_{\mathcal{T}}$), instead, the trend is different. The reason behind the low percentage in these cases is that there are few violations of requirements $\{R_1, R_3, R_4\}$ and $\{R_3, R_4\}$ in $t_{\mathcal{DA}}$ and $t_{\mathcal{T}}$, respectively, so RVC can hardly distinguish configurations in the layers of these requirements.

[3]Note that, as explained at the end of § IV-C, the number of comparisons required to sort all configurations is much lower. However, only for the experiments, we perform all the comparisons to have more detailed information regarding the layers at which configurations are distinguished by HSA.

TABLE III: RQ2–Comparison results of RVC under $\mathcal{TS}$ and every single scenario (*ConsRes* in gray, *InconsRes* in white)

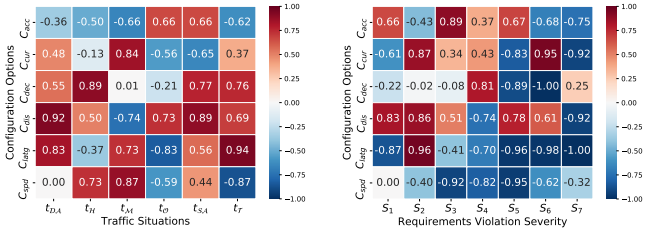
RVCres	$RVCres_{single}$	$t_{\mathcal{SA}}$	$t_{\mathcal{M}}$	$t_{\mathcal{H}}$	$t_{\mathcal{DA}}$	$t_{\mathcal{O}}$	$t_{\mathcal{T}}$
$\boldsymbol{c} \succ \boldsymbol{c}'$	$\boldsymbol{c} \succeq \boldsymbol{c}'$	93.21%	93.45%	94.16%	85.57%	91.14%	83.88%
	$\boldsymbol{c} \prec \boldsymbol{c}'$(I)	0.02%	0.01%	0.09%	0%	0.06%	0.54%
	$\boldsymbol{c} \prec \boldsymbol{c}'$(II)	0.23%	1.33%	0.21%	0.77%	3.14%	1.90%
	$\boldsymbol{c} \prec \boldsymbol{c}'$(III)	1.15%	3.03%	2.78%	10.40%	1.97%	10.73%
	$\boldsymbol{c} \prec \boldsymbol{c}'$(IV)	5.38%	2.18%	2.76%	3.27%	3.69%	2.96%
$\boldsymbol{c} = \boldsymbol{c}'$	$\boldsymbol{c} = \boldsymbol{c}'$	100%	100%	100%	100%	100%	100%
	$\boldsymbol{c} \neq \boldsymbol{c}'$	0%	0%	0%	0%	0%	0%

Answer to RQ1: Compared with CC, HSA is a more feasible approach to distinguish ADS configurations. The distinguishable rate of HSA decreases as the layer in RVC increases from I to IV.

2) Evaluation of RQ2: In RQ1, we observe that the proposed approach HSA is able to distinguish more pairs of configurations, as it is less restrictive in the comparison. Specifically, during the application of RVC in a given layer, the violation severity values across all the scenarios are summed up (Lines 7 and 8 in Alg. 1) and the configuration that overall has a lower value is selected as the better one. Because of this "aggregation" step, there could be scenarios in $\mathcal{TS}$ for which the better configuration is not the one selected by RVC. We here investigate to what extent the configurations selected as safer are indeed optimal for the scenarios in $\mathcal{TS}$.

To do this, we first collect the comparison results computed by the proposed approach. Specifically, for each pair of configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$, we check the result provided by RVC when using the complete set $\mathcal{TS}$ (i.e., *RVCres* in Alg. 1). Moreover, to check how the two configurations $\boldsymbol{c}$ and $\boldsymbol{c}'$ behave on each specific scenario, we also apply RVC repeatedly using a single scenario $t \in \mathcal{TS}$ at a time, and the comparison result is denoted as $RVCres_{single}$. We then calculate the percentage of results of $RVCres_{single}$ (across all the scenarios $\mathcal{TS}$) that correspond to the result of *RVCres* as *ConsRes*, and the percentage of results that do not correspond as *InconsRes*. The higher *ConsRes* is, the better the assessment done by HSA is.

According to Alg. 1, *RVCres* either states that "configuration $\boldsymbol{c}$ is safer than $\boldsymbol{c}'$" (i.e., $\boldsymbol{c} \succ \boldsymbol{c}'$) or that "configuration $\boldsymbol{c}$ ties with $\boldsymbol{c}'$" (i.e., $\boldsymbol{c} = \boldsymbol{c}'$). As shown in Table III, for each traffic situation, we calculate the average of *ConsRes* (in gray cells) and *InconsRes* (in white cells) across all the pairs of configurations. In the case that *RVCres* states $\boldsymbol{c} \succ \boldsymbol{c}'$, *ConsRes* is given by the scenarios of $RVCres_{single}$ in which configuration $\boldsymbol{c}$ is not worse than $\boldsymbol{c}'$ (i.e., $\boldsymbol{c} \succeq \boldsymbol{c}'$), while *InconsRes* is given by the scenarios in which $RVCres_{single}$ states that configuration $\boldsymbol{c}$ is worse than $\boldsymbol{c}'$ (i.e., $\boldsymbol{c} \prec \boldsymbol{c}'$). The



(a) Between configuration options and rankings (b) Between configuration options and requirements violation severity in $t_{\mathcal{O}}$

Fig. 7: RQ3–Spearman correlation (r_s) between configuration options, rankings and requirements violation severity

values of *InconsRes* are reported for the different layers I to IV, to understand where the disagreements between $RVCres$ and $RVCres_{single}$ occur. Instead, in the case that $RVCres$ states $\boldsymbol{c} = \boldsymbol{c}'$, *ConsRes* is given by the scenarios in which $RVCres_{single}$ also claims $\boldsymbol{c} = \boldsymbol{c}'$, while *InconsRes* is given by the scenarios in which $\boldsymbol{c} \neq \boldsymbol{c}'$.

From the results in the table, we can find that the comparison results $RVCres$ are mostly consistent with those of $RVCres_{single}$. The disagreements are mainly in layers III and IV, related to less important requirements. This finding means that the approximation performed by $RVCres$ (i.e., the aggregation of the values of requirements violation severity in the different scenarios) can effectively distinguish safer configurations that are applicable in most scenarios, and may not be optimal only on the less important requirements in a few scenarios. Note that there is no disagreement on $\boldsymbol{c} = \boldsymbol{c}'$: indeed, RVC claims that two configurations are equivalent if they behave the same on all the scenarios.

Answer to RQ2: The approximation performed by RVC is necessary and effective, as the configurations identified as safer are indeed optimal for most of the scenarios.

3) Evaluation of RQ3: We here show how HSA can be used in an industrial setting, and we check whether it provides the engineers of our industrial partner useful insights on how to configure a safer MP in different traffic situations.

To provide suggestions on how to configure a safe MP, we perform two types of analysis. By changing each configuration option, the first one checks how the overall safety ranking is affected in each traffic situation (see § V-B3a), while the second one checks how the violation severity of each requirement is affected in specific traffic situations (see § V-B3b).

Fig. 7(a) and Fig. 7(b) report heat maps describing the correlation between the values of configuration options C_i and safety rankings in $\overrightarrow{\mathcal{C}}$, and configuration options C_i and requirements violation severity S_i, respectively. We use Spearman correlation [26], which is a non-parametric measure to compute monotonic association between two variables [27]. Specifically, Spearman correlation (r_s) ranges from -1 to 1, where $-1 \leq r_s < 0$ (blue cells in the heat maps) means that there is a negative monotonic correlation, and $0 < r_s \leq 1$ (red cells in the maps) means that there is a positive and monotonic correlation; the greater the absolute value of r_s, the stronger the correlation. $r_s = 0$ suggests that there is no correlation. In Fig. 7(a), strong correlation ($|r_s| \to 1$) between C_i and rankings indicates larger effect of the C_i value on the safety rankings in each traffic situation. In Fig. 7(b), strong correlation between C_i and S_i indicates larger effect of C_i value on the satisfaction of R_i in a given traffic situation.

a) Analysis on Overall Safety Rankings: In this analysis, we measure r_s between configuration options and their rankings in the sorted list of $\overrightarrow{\mathcal{C}}$ in six traffic situations. In $\overrightarrow{\mathcal{C}}$, configurations that have smaller rank values are safer (i.e., the top configuration having rank 1 is the safest under HSA). Thus, $r_s > 0$ suggests that increasing the value of configuration option C_i leads to the increment of the ranking position of MP in $\overrightarrow{\mathcal{C}}$ (i.e., more unsafe). As shown in Fig. 7(a), we report a heat map showing the Spearman correlation between configuration options and rankings.

We observe that, for some traffic situations, some configuration options have stronger correlation, while for others the correlation is milder; for example $r_s = 0.94$ for C_{latg} in $t_{\mathcal{T}}$, and $r_s = 0$ for C_{spd} in $t_{\mathcal{DA}}$. When re-configuring MP for a specific traffic situation, ADS engineers should focus on the options having stronger correlation.

We also notice that a given configuration option C_i can have, in different traffic situations, opposite correlations with their safety ranking. For example, the largest gap exists between $t_{\mathcal{O}}$ and $t_{\mathcal{T}}$ for configuration option C_{latg} with $r_s = -0.83$ and $r_s = 0.94$, respectively. The reasons are as follows. In $t_{\mathcal{O}}$ (Fig. 5(e)), when the *ego-vehicle* tries to overtake *vehicle-a* by changing its lane from *lane-1* to *lane-2* and then back to *lane-1*, it should avoid extreme steering angles which may lead the *ego-vehicle* to rollover. If C_{latg} is set very small (meaning that MP does not penalize high lateral acceleration), behaviors like sharp turns are allowed and MP may require to return to *lane-1* suddenly and violate R_1, so possibly leading to rollover. In $t_{\mathcal{T}}$ (Fig. 5(f)), when the *ego-vehicle* is going to turn at the intersection from *lane-1* to *lane-2*, it must avoid the collision with both *vehicle-a* and *vehicle-b*, which requires a large lateral acceleration to perform the turn quickly. However, if C_{latg} is set very large (meaning that MP prefers low lateral acceleration), MP will select a more smooth trajectory (with small lateral acceleration) that may eventually lead to the collision with *vehicle-b* (i.e., violating R_2). Thus, previous observations show that, in order to achieve safety, C_{latg} should be set to a high value in $t_{\mathcal{O}}$, and to a low value in $t_{\mathcal{T}}$. The findings of the opposite correlation between modified configurations and their safety rankings in different traffic situations support our assumption that investigating ADS configurations by traffic situation is needed, as the same configuration can lead to safe or dangerous behaviors in particular traffic situations. Given the results presented in Fig. 7(a), ADS engineers of our industrial partner may consider using different configurations in different traffic situations.

b) Analysis on Requirements Violation Severity: We provide another analysis showing how, in a given traffic situation, requirements satisfaction is affected by the modification of

different configuration options. Let us consider $t_{\mathcal{O}}$ as an example. Fig. 7(b) reports a heat map showing the r_s between configuration options and requirements violation severity; heat maps for the other traffic situations are provided online [18].

We observe that the modification of some configuration option affects the satisfaction of some requirements more than that of other configuration options. For example, modifying C_{latg} affects more $\{R_5, R_6, R_7\}$ that are directly related to lateral acceleration (in particular R_7 with $r_s = -1$), while it affects less requirement R_3 ($r_s = -0.41$) on speed limit which is less related to lateral acceleration. In some cases, the correlation between the configuration option and the requirement could be due to some specific characteristic of the traffic situation. For example, C_{latg} is also strongly positively correlated with R_2 on safety distance ($r_s = 0.96$), meaning that having a lower value of C_{latg} (i.e., allowing higher lateral acceleration) leads to less violation of R_2; indeed, having a higher lateral acceleration in this traffic situation, allows the ego vehicle to turn faster and keep a safe distance with *vehicle-b* coming from behind. However, as explained in § V-B3a, for the overall safety, it is still better to have a high value for C_{latg}, because this guarantees more stability of the vehicle, which is a more important requirement. In any case, this more detailed analysis allows the ADS engineers of our industry partner to better understand which kind of trade-off exists among the requirements satisfaction.

Answer to RQ3: In an industrial setting, HSA can be leveraged to analyze the correlations between ADS configuration options and safe driving behaviors, showing that HSA is useful for configuring a safe ADS.

C. Threats to Validity

We discuss possible threats to the approach validity [28].

Conclusion validity. To mitigate the conclusion validity risks caused by randomly generated configurations, we generate 60 alternative configurations with single-weight modification. In this way, ADS engineers can assess the effect of every single configuration option on ADS safety. However, we acknowledge that there could be a combined effect of multiple options on the ADS safety; therefore, as future work, we plan to assess the approach also under multiple modifications of the configuration options.

Internal validity. Faults in HSA implementation can bias the results; to mitigate this threat, we carefully inspected the implementation and checked that RVA results are consistent with the driving behaviors of MP.

External validity. These threats include the degree to which the subject ADS, and selected traffic situations are representative. Our evaluation is conducted with only one system MP, to solve the specific problems encountered by our industrial partner during the testing and configuration of MP; so by definition, the results are specific to that system [29]. However, also other ADSs (e.g., Apollo framework [30] and Autonomoose framework [31]) provide parameters for configuring their decision systems. As future work, we plan to assess whether HSA is also applicable to these ADSs. Moreover, the HSA may not be generalizable to different driving situations. Nevertheless, we have minimized the risk of this threat by evaluating it in six different traffic situations. In future work, our goal is to have a complete set of all traffic situations by following the principle of the environment modeling-based paradigm [32] which allows us to investigate more traffic situations.

VI. Related Work

A. Safety Assessment for ADS

There is an urgent societal need to assess whether ADSs are safe enough [33]. An autonomous vehicle normally is evaluated either on real roads or in simulated environments. On-road testing is necessary and widely used in industrial ADSs such as Waymo [34] and Voyage.auto [35]. However, there is a lack of evaluation measures for ADS safety in this type of testing, and the only safety guarantees are provided by the observation that the car drove "over x million kilometers without collision" [36], [37]. Recently, the time-consuming and costly on-road testing is augmented by virtual testing in computer simulations [38]–[47] aiming at generating critical test scenarios in which the autonomous vehicle may crash. Test engineers use simple evaluation measures like time to collision [38], [39], [48] and safe distance [40], [42], [43] to find dangerous scenarios. However, safety in ADSs is a complex and interdisciplinary matter that requires more sophisticated metrics coming from requirements engineering. To narrow the gap between system developers' knowledge about safety and the safety concerns of requirements engineers, we consider the possibility of quantitative assessment for multiple safety requirements. We design the Safety Metric for each requirement and propose RVA to quantitatively assess the violation severity for elicited safety requirements. In addition, we prioritize the safety-related requirements according to the criticality in case of violations; such prioritization makes it possible to assess the safety of ADS configurations under multiple safety-related requirements in RVC.

B. ADS Configuring

ADSs are highly configurable systems with many configuration parameters that may interact and ultimately form an exponentially large configuration space [49]. In the decision-making process of the ADS control software, trade-offs are configurable and may lead to different results of requirements violations. Recent approaches aim at finding *avoidable collision* scenarios in which the ego vehicle does not collide with an alternative configuration [11]. In addition, an approach is proposed to investigate the relationship between the ADS configurations and the reason for collisions during driving [9]. However, these approaches only consider the reconfiguration of ADS for one specific safety requirement (i.e., safe distance), and none of them consider the problem of trade-offs among multiple safety requirements. Indeed, configurations chosen for keeping a safe distance may sacrifice the satisfaction of other requirements, e.g., vehicle stability. In contrast, we take into account multiple safety-related requirements and devise a

novel method to assess their violations in given traffic situations. With HSA, ADS engineers can analyze the differences in safety requirements violations for ADS configurations from a systematic perspective. Thus, HSA may help to distinguish safer configuration for the ADS control software.

VII. Conclusion

In this paper, we have proposed HSA, a hierarchical assessment of safety requirements for configurable ADSs. HSA builds on requirements violation analysis (RVA) and requirements violation comparison (RVC), which is guided by prioritized requirements by importance levels to distinguish safer configurations. HSA is evaluated on an industrial ADS. We demonstrated the effectiveness of HSA to distinguish safer ADS configurations and present the usage of HSA to configure safer ADS in an industrial setting. In the future, we plan to (1) apply HSA to thoroughly investigate safe configurations in a complete set of traffic situations, and (2) extend HSA to support the safe configuring of more complex ADSs.

References

[1] K. Czarnecki, "Requirements engineering in the age of societal-scale cyber-physical systems: The case of automated driving," in *Proc. 26th IEEE International Requirements Engineering Conference (RE)*, 2018, pp. 3–4, doi: 10.1109/RE.2018.00-57.

[2] Y. Luo, Y. Yu, Z. Jin, and H. Zhao, "Environment-centric safety requirements for autonomous unmanned systems," in *Proc. 27th IEEE International Requirements Engineering Conference (RE)*, 2019, pp. 410–415, doi: 10.1109/RE.2019.00054.

[3] K. Czarnecki, "Automated driving system (ADS) high–level quality requirements analysis– driving behavior safety," *Waterloo Intelligent Systems Engineering Lab (WISE) Report, University of Waterloo*, 2018.

[4] Guardian, "Self-driving uber kills arizona woman in first fatal crash involving pedestrian," https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe, 2018.

[5] Forbes, "Tesla in Taiwan crashes directly into overturned truck, ignores pedestrian, with autopilot on," https://www.forbes.com/sites/bradtempleton/2020/06/02/tesla-in-taiwan-crashes-directly-into-overturned-truck-ignores-pedestrian-with-autopilot-on, 2020.

[6] S. M. Ågren, E. Knauss, R. Heldal, P. Pelliccione, G. Malmqvist, and J. Bodén, "The manager perspective on requirements impact on automotive systems development speed," in *Proc. 26th IEEE International Requirements Engineering Conference (RE)*, 2018, pp. 17–28, doi: 10.1109/RE.2018.00-55.

[7] T. Gu, J. M. Dolan, and J. Lee, "Runtime-bounded tunable motion planning for autonomous driving," in *Proc. 7th IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1301–1306, doi: 10.1109/IVS.2016.7535558.

[8] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *CoRR*, vol. abs/1708.06374, 2017.

[9] X. Zhang, P. Arcaini, F. Ishikawa, and K. Liu, "Investigating the configurations of an industrial path planner in terms of collision avoidance," in *Proc. 31st IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 301–312, doi: 10.1109/ISSRE5003.2020.00036.

[10] E. Onieva, U. Hernández-Jayo, E. Osaba, A. Perallos, and X. Zhang, "A multi-objective evolutionary algorithm for the tuning of fuzzy rule bases for uncoordinated intersections in autonomous driving," *Inf. Sci.*, vol. 321, pp. 14–30, 2015, doi: 10.1016/j.ins.2015.05.036.

[11] A. Calò, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Generating avoidable collision scenarios for testing autonomous driving systems," in *Proc. 13th IEEE International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 375–386, doi: 10.1109/ICST46399.2020.00045.

[12] R. McAllister, Y. Gal, A. Kendall, M. van der Wilk, A. Shah, R. Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning," in *Proc. 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 4745–4753, doi: 10.24963/ijcai.2017/661.

[13] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, 2016, doi: 10.1109/TIV.2016.2578706.

[14] Y. Luo, X.-Y. Zhang, P. Arcaini, Z. Jin, H. Zhao, F. Ishikawa, R. Wu, and T. Xie, "Targeting requirements violations of autonomous driving systems by dynamic evolutionary search," in *Proc. 36th IEEE/ACM International Conference on Automated Software Engineering*, 2021, pp. 279–291, doi: 10.1109/ASE51524.2021.00034.

[15] P. Jamshidi, J. Cámara, B. R. Schmerl, C. Kästner, and D. Garlan, "Machine learning meets quantitative planning: enabling self-adaptation in autonomous robots," in *Proc. 14th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2019, pp. 39–50.

[16] V. Klös, T. Göthel, and S. Glesner, "Runtime management and quantitative evaluation of changing system goals in complex autonomous systems," *J. Syst. Softw.*, vol. 144, pp. 314–327, 2018, doi: 10.1016/j.jss.2018.06.076.

[17] K. Neace, R. A. Roncace, and P. Fomin, "Goal model analysis of autonomy requirements for unmanned aircraft systems," *Requir. Eng.*, vol. 23, no. 4, pp. 509–555, 2018, doi: 10.1007/s00766-017-0278-6.

[18] HSA, "Supplementary materials for "Hierarchical Assessment of Safety Requirements for Configurations of Autonomous Driving Systems"," 2022. [Online]. Available: https://figshare.com/s/5c88f6fb980c99ab6378

[19] N. Riegel and J. Dörr, "A systematic literature review of requirements prioritization criteria," in *Proc. 21st International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, vol. 9013. Springer, 2015, pp. 300–317, doi: 10.1007/978-3-319-16101-3_22.

[20] R. Edwards and N. Bencomo, "DeSiRE: further understanding nuances of degrees of satisfaction of non-functional requirements trade-off," in *Proc. 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS@ICSE)*, 2018, pp. 12–18, doi:10.1145/3194133.3194142.

[21] E. M. Fredericks, B. DeVries, and B. H. C. Cheng, "AutoRELAX: automatically RELAXing a goal model to address uncertainty," *Empir. Softw. Eng.*, vol. 19, no. 5, pp. 1466–1501, 2014, doi: 10.1007/s10664-014-9305-0.

[22] A. Arcuri, "It really does matter how you normalize the branch distance in search-based software testing," *Softw. Test. Verification Reliab.*, vol. 23, no. 2, pp. 119–147, 2013, doi: 10.1002/stvr.457.

[23] T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, and S. A. Seshia, "Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems," in *Proc. 31st International Conference on Computer Aided Verification (CAV)*, ser. Lecture Notes in Computer Science, vol. 11561, 2019, pp. 432–442, doi: 10.1007/978-3-030-25540-4_25.

[24] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1821–1827, doi: 10.1109/IVS.2018.8500406.

[25] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Computing Surveys*, vol. 45, no. 1, pp. 11:1–11:61, 2012, doi: 10.1145/2379776.2379787.

[26] C. Spearman, "The proof and measurement of association between two things," *American Journal of Psychology*, vol. 15, pp. 88–103, 1904.

[27] J. Hauke and T. Kossowski, "Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data," *Quaestiones geographicae*, vol. 30, no. 2, pp. 87–93, 2011.

[28] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012.

[29] L. C. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "The case for context-driven software engineering research: Generalizability is overrated," *IEEE Software*, vol. 34, no. 5, pp. 72–75, 2017, doi: 10.1109/MS.2017.3571562.

[30] Baidu, "Apollo," https://github.com/ApolloAuto/apollo, 2022.

[31] M. Antkiewicz, M. Kahn, M. Ala, K. Czarnecki, P. Wells, A. Acharya, and S. Beiker, "Modes of automated driving system scenario testing: Experience report and recommendations," *SAE International Journal of Advances and Current Practices in Mobility*, vol. 2, no. 2020-01-1204, pp. 2248–2266, 2020, doi: 10.4271/2020-01-1204.

[32] Z. Jin, *Environment Modeling based Requirements Engineering for Software Intensive Systems.* Morgan Kaufmann, 2018.

[33] X. Zhao, V. Robu, D. Flynn, K. Salako, and L. Strigini, "Assessing the safety and reliability of autonomous vehicles from road testing," in *Proc. 30th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 13–23, doi: 10.1109/ISSRE.2019.00012.

[34] W. Inc., "On the Road to Fully Self-Driving: Waymo Safety Report," 2018.

[35] V. Inc., "Open Autonomous Safety," 2019.

[36] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?" *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016.

[37] H. Winner, S. Hakuli, F. Lotz, and C. Singer, *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2015.

[38] R. Ben Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing advanced driver assistance systems using multi-objective search and neural networks," in *Proc. 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 63–74, doi: 10.1145/2970276.2970311.

[39] ——, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proc. 40th IEEE/ACM International Conference on Software Engineering (ICSE)*, 2018, pp. 1016–1026, doi: 10.1145/3180155.3180160.

[40] R. Ben Abdessalem, A. Panichella, S. Nejati, L. C. Briand, and T. Stifter, "Testing autonomous cars for feature interaction failures using many-objective search," in *Proc. 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2018, pp. 143–154, doi: 10.1145/3238147.3238192.

[41] A. Gambi, M. Mueller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proc. 28th ACM International Symposium on Software Testing and Analysis (ISSTA)*, 2019, pp. 318–328, doi: 10.1145/3293882.3330566.

[42] G. Li, Y. Li, S. Jha, T. Tsai, M. B. Sullivan, S. K. S. Hari, Z. Kalbarczyk, and R. K. Iyer, "AV-FUZZER: finding safety violations in autonomous driving systems," in *Proc. 31st IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2020, pp. 25–36, doi: 10.1109/ISSRE5003.2020.00012.

[43] T. Laurent, P. Arcaini, F. Ishikawa, and A. Ventresque, "Achieving weight coverage for an autonomous driving system with search-based test generation," in *Proc. 25th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2020, pp. 93–102, doi: 10.1109/ICECCS51672.2020.00018.

[44] Y. Li, J. Tao, and F. Wotawa, "Ontology-based test generation for automated and autonomous driving functions," *Information and Software Technology*, vol. 117, 2020, doi: 10.1016/j.infsof.2019.106200.

[45] F. Hauer, T. Schmidt, B. Holzmüller, and A. Pretschner, "Did we test all scenarios for automated and autonomous driving systems?" in *Proc. 22nd IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2950–2955, doi: 10.1109/ITSC.2019.8917326.

[46] C. Zhang, Y. Liu, D. Zhao, and Y. Su, "Roadview: A traffic scene simulator for autonomous vehicle simulation testing," in *Proc. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 1160–1165, doi: 10.1109/ITSC.2014.6957844.

[47] D. Zhao and H. Peng, "From the lab to the street: Solving the challenge of accelerating automated vehicle testing," *CoRR*, vol. abs/1707.04792, 2017.

[48] S. Noh, "Probabilistic collision threat assessment for autonomous driving at road intersections inclusive of vehicles in violation of traffic rules," in *Proc. 31st IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2018, pp. 4499–4506, doi: 10.1109/IROS.2018.8593645.

[49] A. Calò, P. Arcaini, S. Ali, F. Hauer, and F. Ishikawa, "Simultaneously searching and solving multiple avoidable collisions for testing autonomous driving systems," in *Proc. 22nd ACM Genetic and Evolutionary Computation Conference (GECCO)*, 2020, pp. 1055–1063, doi: 10.1145/3377930.3389827.