Machine Learning-based Fault Injection for Hazard Analysis and Risk Assessment*

Bentley James Oakes[0000−0001−7558−1434], Mehrdad Moradi**[0000−0001−8748−069X], Simon Van Mierlo[0000−0002−4043−6883], Hans Vangheluwe[0000−0003−2079−6643], and Joachim Denil[0000−0002−4926−6737]

University of Antwerp and Flanders Make vzw, Belgium
{bentley.oakes, mehrdad.moradi, simon.vanmierlo, hans.vangheluwe, joachim.denil}@uantwerpen.be

Abstract. Current automotive standards such as ISO 26262 require Hazard Analysis and Risk Assessment (HARA) on possible hazards and consequences of safety-critical components. This work attempts to ease this labour-intensive process by using machine learning-based fault injection to discover representative hazardous situations. Using a Simulation-Aided Hazard Analysis and Risk Assessment (SAHARA) methodology, a visualisation and suggested hazard classification is then presented for the safety engineer. We demonstrate this SAHARA methodology using machine learning-based fault injection on a safety-critical use case of an adaptive cruise control system, to show that our approach can discover, visualise, and classify hazardous situations in a (semi-)automated manner in around twenty minutes.

Keywords: Hazard Analysis · Risk Assessment · Verification · Fault injection · Reinforcement Learning · Signal Temporal Logic.

1 Introduction

Automotive systems are complex *cyber-physical systems* with ever-tightening safety and production efficiency requirements. These qualities must be ensured even as automotive software contains lines of code numbering in the tens of millions [6] to support many modern features, including autonomous operation.

The well-known ISO 26262 standard mandates manufacturers to perform safety and hazard analysis of their vehicles [11]. In particular, *hazards* and *faults* must be shown to be adequately considered and handled by the manufacturer in the form of supporting evidence cases. ISO 26262 defines one outcome of this analysis as the Automotive Safety Integrity Level (ASIL), which denotes the risk of a hazard and, therefore, the level of risk reduction required to be

* This work was partly funded by Flanders Make vzw, the strategic research centre for the Flemish manufacturing industry; and by the aSET project (grant no. HBC.2017.0389) of the Flanders Innovation and Entrepreneurship agency (VLAIO).
** Corresponding author. Tel: +32 487 850 695.

implemented by automotive components. Performing this *Hazard Analysis and Risk Assessment* (HARA) is manual work that requires hours of discussions between safety engineers [16]. Automating aspects of this analysis, therefore, greatly reduces the time taken to understand hazardous situations.

A *Simulation-Aided Hazard Analysis and Risk Assessment* (SAHARA) approach can utilise intelligent Fault Injection (FI) on vehicle components, which are then simulated for the visualisation and classification of hazardous situations [21]. Further described in Section 3, this SAHARA methodology[1] involves models of the components under study, information on how to inject faults, the safety-critical scenarios of interest, and a process to suggest a hazard classification for the simulation result. This (semi-) automated methodology provides safety engineers with representative visualisations and safety classifications of a system's faulty behaviour in various hazardous situations.

However, the SAHARA methodology defined in [21] presents few details on the FI procedure and implementation. For example, selecting the injection site and optimising the fault parameters is not discussed, and no indication of the performance of the FI procedure is given.

Our research focuses on intelligent FI, which employs a Reinforcement Learning (RL) algorithm to discover fault parameters [18]. RL is a large category of Machine Learning (ML) algorithms learning from the environment by interacting dynamically with it [20]. RL is used to automatically identify the parameters for critical faults that should be injected into the component under test to provoke increasingly more hazardous behaviour. In this work, we place our FI approach within the SAHARA methodology and inject hazardous faults. The CARLA open-source simulator for automotive research [8] then produces a visualisation of the resulting system behaviour, and an ASIL is suggested using temporal logic on the simulation traces.

This paper's contributions are therefore: (i) detailing how the ML-based FI process uses the available data within the SAHARA methodology to automatically produce hazardous situations, (ii) providing an example of the FI and SAHARA processes on a use case, including an indication of the approach performance, and (iii) a discussion of the benefits and drawbacks of placing ML-based FI within the SAHARA process.

Section 2 introduces the adaptive cruise control example. Section 3 describes FI within the SAHARA methodology, while Section 4 an indication of performance. Section 5 discusses the approach, while related literature is presented in Section 6 and Section 7 concludes the paper and describes future work.

2 Adaptive Cruise Control

This section introduces the Adaptive Cruise Control (ACC) system under study and the potential hazards that may arise from faults in the component.

[1] Other approaches such as [13, 14] refer to a SAHARA approach. This paper uses SAHARA solely to refer to the methodology of [21].

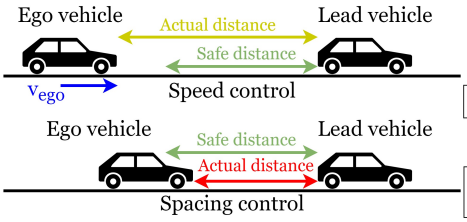

Fig. 1: ACC *speed* and *spacing* modes.

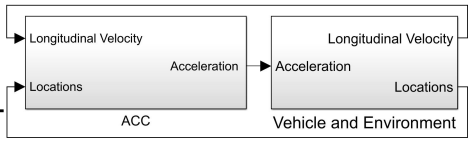

Fig. 2: ACC model in Simulink®.

The purpose of the ACC is to regulate the speed of the *ego vehicle* (the vehicle of concern) to ensure that it does not approach the rear of any *lead vehicles* too closely. The user of the ACC defines a *preferred speed* and *safe distance*, which is compared to the *relative distance* between the ego and lead vehicles. Figure 1 presents the two modes of the ACC. The first mode of the ACC is the *speed control* mode; the ACC directs the ego vehicle to increase the vehicle's speed to the preferred speed, potentially decreasing the relative distance. The second *spacing control* mode occurs when the ego vehicle is within safe distance to the lead vehicle. Here, the ACC directs the ego vehicle to reduce its speed such that the safe distance is maintained.

The Simulink® model used for this work is shown in Fig. 2, where the *ACC* is modelled in the left-hand block and the *ego vehicle dynamics and environment* are modelled in the right-hand block[2]. In this work, the automotive simulator CARLA [8] will replace the vehicle dynamics and environment block.

As the ACC component can control the vehicle's acceleration and the resulting distance from other vehicles, the ACC is a safety-critical component. Even a minor fault could violate the safety requirement that the relative distances between vehicles is greater than a set safe distance. In a more hazardous situation, a major fault in the ACC could lead to an unintended acceleration into the rear of the lead vehicle, potentially resulting in severe injuries or death. Therefore, this ACC component must be intensively examined in a structured manner to determine possible hazards and their consequences.

In Section 3.3, we inject a fault in the longitudinal velocity of the ego vehicle, which is transferred to the ACC such that the ACC does not accurately know the vehicle's speed. This hazardous situation can lead to unintended acceleration as visualised and classified with the SAHARA methodology.

3 ML-based FI within the SAHARA Methodology

The Simulation-Aided Hazard Analysis and Risk Assessment (SAHARA) methodology focuses on assisting with (semi-) automated reasoning about the hazards

[2] The model is an adapted version of https://www.mathworks.com/help/mpc/ug/adaptive-cruise-control-using-model-predictive-controller.html.

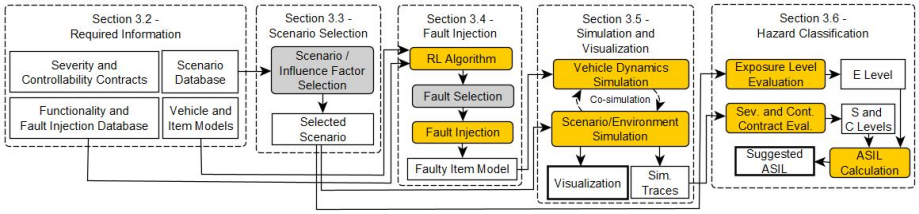

Fig. 3: The overall SAHARA architecture and workflow (adapted from [21]). Yellow blocks are automatic actions, and gray blocks are manual actions.

and risks present in a safety-critical system [21]. In summary, this methodology utilises specifications of scenarios, faults, and vehicle dynamics, which are combined and fed as input into simulations, which in turn provide data for visualisations and classification of the (potentially) hazardous situation.

The following sections will address applying the five prominent components of the approach as indicated in Fig. 3: *required information*, *scenario selection*, *fault injection and reinforcement learning algorithm*, *simulation and visualisation*, and *classification of hazard level*.

3.1 Required Information

Scenario Database. This scenario information combines the map and path information for vehicles, along with different *influence factors*, such as relevant characteristics of the vehicle, road conditions, and vehicle/pedestrian interactions [21, 13].

Vehicle and Item Models. The SAHARA methodology requires detailed vehicle models, including the dynamics and the component(s) under study. Faults are injected into these component models, and the dynamics models are then used to simulate a scenario and assess the safety of the faulty component. In this work, we utilise the simple vehicle dynamics model built into CARLA, and the ACC model available within the Simulink documentation (see Section 2).

Functionality and Fault Database. Information on which component is under study and how it may fail is also necessary for utilising the SAHARA methodology, such that faults can be appropriately applied to the item models as discussed in Section 3.3. The functionality under study in this work is the reporting of the ego vehicle's longitudinal velocity to the ACC (see Section 2), with *sensor noise* and *stuck-to* faults available ([21]).

Severity and Controllability Contracts. The SAHARA methodology requires contracts to suggest levels of severity and controllability as defined by the ISO 26262 standard. These contracts are discussed in detail in Section 3.5.

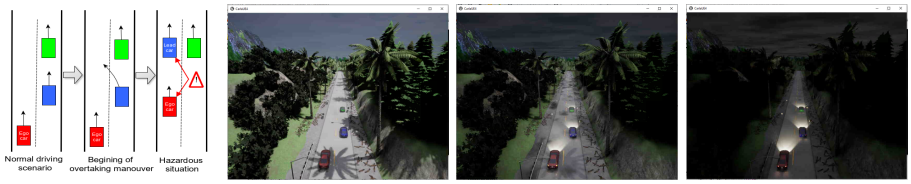

Fig. 4: Base scenario, and visualized as *DryRoad*, *RainyRoad*, and *NightRoad*.

3.2 Scenario Selection

An analysis of automotive hazards involves reasoning about the safety of a component in various scenarios. These scenarios involve the layout of the roads, the road surface, as well as effects like the weather. These factors may all impact the dynamics or controllability of the vehicle and thus must be considered in a safety assessment process.

As envisaged in the SAHARA methodology [21], the safety engineer would select representative scenarios of interest through a tabular scenario description file. This work selects three scenarios: a straight road in clear weather, rainy weather, and night-time rainy weather, as shown in Fig. 4. The rainy weather affects both the visuals of the situation and the friction parameters of the road[3]. The night-time scenario does not affect the vehicle dynamics, but would make the situation more hazardous for a human driver.

Each scenario represents the same driving manoeuvre as shown on the left-hand side of Fig. 4. The red vehicle at the bottom is the ego vehicle, which must detect the blue middle vehicle moving into the left lane to overtake the furthest green vehicle. As explained in Section 2, the ACC must function appropriately on the ego vehicle to detect this movement into its lane, decrease the ego vehicle's speed if necessary, and avoid an accidental rear-end collision.

3.3 Fault Injection and Reinforcement Learning Algorithm

Fault injection (FI) is a well-known technique that exposes the system to a fault to allow the test engineer to understand if the system can adequately respond or whether further design changes are required. Our approach focuses on the most common *stuck-to-value* fault type in the sensors (input values) of components [25].

For example, a *stuck-to-value* fault may force a signal's value to be intermittently 'stuck' to a certain value at some simulation time, as modelled in Fig. 5. The middle block is a 'switch' block, which changes the output from the regular input I to the faulty value V at all timesteps after time T. However, this fault must still be parameterised to answer a) *where* to inject this fault, b) *when* the switch should occur, and c) *what* the faulty value should be [3].

[3] Friction values sourced from Fig. 24 of Singh and Taheri [28].

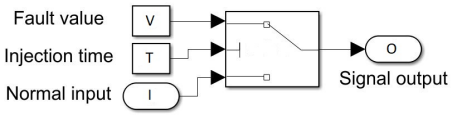

Fig. 5: *Stuck-to-value* fault at time T.

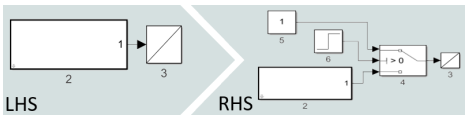

Fig. 6: Model transformation FI [19].

Fault Injection in the Adaptive Cruise Control. The ACC controls the ego vehicle's acceleration based on the information from incoming radar combined with the ACC's information on the longitudinal velocity of the ego vehicle (Section 2). As in [18], this work studies the presence of a stuck-to-value fault in the *ego longitudinal velocity* sensor of the ACC as seen in the top-left of Fig. 2. That is the sensor which reports the current velocity of the ego vehicle to the ACC. With this fault, the ACC will have incorrect knowledge of the ego vehicle's velocity, potentially resulting in a hazardous situation or collision.

In the SAHARA methodology, the safety engineer would select the longitudinal velocity signal as the fault location and the stuck-to-value fault from the ACC functionality and fault database. This FI is then performed using a framework to perform rule-based transformations on Simulink models [7, 19]. Patterns utilising Simulink blocks can be matched in a model, and a rewrite pattern can then add, remove, or modify blocks. In Fig. 6 the left-hand side of the rule is the block pattern to match and the right-hand side is the replacement pattern.

Use of Machine Learning for Fault Injection. As a contribution to the SAHARA methodology, this work integrates the machine learning-based FI approach from previous work [18]. In this approach, a Reinforcement Learning (RL) framework searches the parameter space of the injected faults (*when* and *what value* to inject) over several simulations to force hazardous situations.

The framework utilises domain knowledge to set the boundaries and steer the direction of the parameter search for the RL agent. For example, the reward function in RL includes three parameters: the *time* of the simulation, the *velocity* of the ego vehicle, and the *relative distance* of the ego and lead vehicles. This reward function steers the search for fault values towards those that increase the velocity while decreasing the time until the collision occurs and the relative distance. This thus provokes as serious a crash as possible.

The FI framework runs multiple iterations of simulations with the fault parameters tuned each time to reach increasing reward function values corresponding to more hazardous situations. This results in a set of fault parameters to inject into the scenario to provoke the most hazardous behaviour found (as defined by reward value), and explore the most relevant safety consequences of faults.

Note that the RL framework must simulate the vehicle's behaviour in the scenario to determine the outcome. Currently, this simulation is performed 'headless' (without visualisation) within Simulink to avoid any overhead. These optimisation simulations are separate from the simulation required for visualisation as discussed in the next section.

3.4 Simulation and Visualisation

At this step in the SAHARA methodology, the appropriate scenario and fault (parameterised using RL) have been selected, and the fault has been injected into the component model. The next stage is the simulation of the scenario to produce a) a visualisation for use by experts in a safety assessment process, and b) traces for a preliminary safety classification.

As in [30], we select the open source CARLA simulator [8] for its high-quality visualisations, easy integration with other tools such as Simulink and Python, and default vehicle dynamics model. The scenarios are loaded into CARLA by modifying the weather and time of day on a built-in map.

The scenario is a co-simulation between CARLA and Simulink using a Python bridge to synchronise each time step. This simulation runs for a predetermined time as set in the scenario parameters. Simulink simulates the ACC, while CARLA simulates the vehicle dynamics and environment. Simulation traces are also produced for the assessment of hazardous situation. These signals include the *acceleration*, *velocity*, and a measure of the *collision impulse* of the ego vehicle (with any kind of object), the *relative velocity and distance* to the lead vehicle, and a measure of the *time gap* before a collision would occur ([26]).

From the simulation, CARLA also produces a visualisation for the safety engineer to utilise to reason about the consequences of the studied fault. These visualisations thus show to the safety engineer a sense of the driver's experience as well as possible outcomes of the failure of the component. This provides insight into how faulty behaviour could be hazardous.

3.5 Hazard Classification

The last step of the SAHARA methodology is to examine the resulting simulation traces to suggest a preliminary classification of the scenario hazard level.

The ISO 26262 standard specifies that an Automotive Safety Integrity Level (ASIL) be produced for a particular scenario by classifying the *exposure* (E), *severity* (S), and *controllability* (C) level of the hazardous scenario [11]. The ASIL is then provided by lookup in a table taking the E, C, and S into account to give ASIL QM (lowest) or A to D (highest). The ASIL, therefore, provides a guide to the safety-critical nature of a component's faults.

Exposure - The *exposure* (E) level of the scenario estimates the likelihood of the scenario from a scale from E1 (low exposure) to E4 (high exposure). The literature presents an automated approach to this calculation based on the probability of each influence factor in the scenario [13].

Severity - The *severity* (S) level concerns the potential injuries or death caused in the scenario, ranging from S1 (no injuries) to S4 (multiple severe injuries or deaths).

Controllability - The *controllability* (C) level of a situation also ranges from C1 to C4. This level represents the difficulty in controlling or avoiding the situation. This includes the driver of the vehicle as well as other participants in the scenario, such as pedestrians.

```
Contract sahara_sev_s2_a{
  description "S2: col_accel >= 40 G for 0.1"
  statements{
    Property col_accel :=
      div(collImpulse, constant CI_SCALE) >=
          val 40 gForce
  }
  scope Globally
  pattern Existence: col_accel occurs-
      eventually for-duration 0.1 s
  action Severity == S2
  generate-STL
}
```

Fig. 7: *S2 Severity* contract.

```
Contract sahara_con_rt2{
  description "Time gap between vehicles is
      [0.5 1.3) seconds"
  statements{
    Property short_time_gap :=
    timeGap in Interval [ min 0.5 max 1.3
        unit second)
  }
  scope Globally
  pattern Existence: short_time_gap occurs-
      eventually
  action Controllability == C2
  generate-STL
}
```

Fig. 8: *C2 Controllability* contract.

Table 1: *Severity* and *Controllability* contracts for hazard classification.

S. Level	Conditions	C. Level	Conditions
S0	$collImpulse = 0$ G	C0	$accel$ in $\pm 1.47 m/s^2$
S1	$collImpulse \geqq 0.01$ G		$timeGap > 2.6$ s
S2	$collImpulse \geqq 40$ G for 0.1 s	C1	$accel$ in $\pm 3.07 m/s^2$
	$collImpulse \geqq 25$ G for 0.2 s		$timeGap$ in 1.3 to 2.6 s
	$collImpulse \geqq 15$ G for 0.6 s	C2	$accel > \pm 3.07 m/s^2$
S3	$collImpulse \geqq 100$ G for 0.01 s		$timeGap$ in 0.5 to 1.3 s
	$collImpulse \geqq 50$ G for 0.04 s	C3	$timeGap < 0.5$ s
	$collImpulse \geqq 45$ G for 0.1 s		
	$collImpulse \geqq 30$ G for 0.3 s		
	$collImpulse \geqq 25$ G for 0.8 s		

Controllability and Severity Contracts. The SAHARA methodology proposes assigning severity and controllability levels by developing temporal logic contracts that operate over the traces output from a simulation [21]. In this work, we further develop the provided contracts in Signal Temporal Logic (STL) [15] and apply them to the simulation traces produced by CARLA. Temporal logic is utilised due to the requirement to reason about both the value of signals as well as the temporal duration of conditions, such as 'at least 0.4 seconds'.

Severity Contracts - The ego vehicle collision impulse experienced in a collision can be used as a proxy for the severity level experienced [21]. Table 1 includes the assignment of collision impulse ranges to severity levels specified by [21] utilising data from [27]. For example, if the collision impulse is experienced at over 40 Gs of force for at least 0.1 seconds, then $S2$ is assigned by Fig. 7.

Controllability Contracts - Two factors are included as proxies for controllability in Table 1: a) the longitudinal acceleration of the ego vehicle, where extreme values indicate a more aggressive driver [2], and b) a measure of reaction time for a driver in case of a lead vehicle emergency brake. That is, how many seconds the driver has to respond before a collision [26]. For example, Fig. 8 assigns *C2* (moderately uncontrollable) if the time gap between the vehicles is less than a standard reaction time of 1.3 seconds [5].

Contract Evaluation. For verification, each contract in Table 1 is mapped to an equivalent STL representation [4]. This mapping process increases the usability of the contract verification approach, as STL can be difficult to reason about and write by hand. Instead, this domain-specific contract language allows for the specification of contracts using familiar operators and units.

For example, Eq. (1) shows the STL generated for the *Severity* contract seen in Fig. 7. This STL converts the collision impulse into the correct unit and implements the *Existence* pattern with duration present in Fig. 7. The resulting STL is then verified against the vehicle dynamics traces produced by CARLA.

$$\textbf{eventually}(\textbf{always}[0:0.1]((collImpulse/14.0) \geqq 392.0)) \tag{1}$$

These traces and the contract STL are fed as input into the Python-based RTAMT verification library[4]. Each contract's STL is checked in turn in an offline manner on the simulation traces. If the specification succeeds, then the trace (and therefore the situation) is assigned at least that severity or controllability level.

Outcome. As discussed in the SAHARA methodology and Section 3.2, the exposure (E) level can be determined by examining the influence factors of the scenario. The severity (S) and controllability (C) levels are then calculated by verifying temporal logic contracts on the resulting simulation trace. The resulting ASIL is then determined through a lookup table of the E, S, and C levels.

This ASIL suggestion is presented to the safety engineer along with a visualisation of the fault scenario to assist in assessing the hazardous situation. Even though these artifacts are only a suggestion of possible outcomes, they may provide insight into the hazardous nature of the situation.

4 Results

Table 2 displays the results of our application of the SAHARA methodology with ML-based FI to the ACC use case. For each scenario, the calculated *Severity* and *Controllability* levels and a suggested ASIL are presented. As each of these scenarios is quite likely, an exposure level of *E4* is statically assigned. Visualisations are also available online for all regular and faulty scenarios[5].

For the non-faulty scenarios, the simulation traces indicate that no collision occurred. Therefore the severity level is the lowest, and the ASIL remains as *Quality Management* (QM). In the *RainyRoad* scenario, the stopping distance and the *timeGap* between the vehicles are modified due to the lowered road friction. The contracts from Table 1 thus assign a higher *Controllability* level (less controlled).

For the scenarios with faults, a collision is provoked by the machine learning approach in Section 3.3. There is a significant collision impulse as the velocity

[4] https://github.com/nickovic/rtamt

[5] https://www.youtube.com/playlist?list=PLNyNvnuIvPKvsmUT1I-hwEYDMyZ7YGZkA

Table 2: Suggested hazard classifications and visualisations for all scenarios.

Scenario	**Without Fault**			**With Fault**		
	DryRoad	*RainyRoad*	*NightRoad*	*DryRoad*	*RainyRoad*	*NightRoad*
S Level	0	0	0	3	3	3
C Level	1	2	2	3	3	3
ASIL	QM	QM	QM	D	D	D

sensor of the ego vehicle becomes faulty as the lead vehicle is overtaking. The rapid acceleration then causes an impact between the vehicles that could cause grave injury. Thus, a severity level of *S3* is assigned, leading to a suggested hazard classification of ASIL D (the highest level).

The simulation time in our approach is divided into three parts: a) injecting faults into the ACC Simulink model takes around 2.7 minutes, b) The RL process then operates on this faulty model and finds multiple parameter sets (fault amplitude and injection time) causing hazardous situations in about 13.5 minutes, and c) For each one of these critical parameter sets found, simulation and visualisation takes around 2.5 minutes with three seconds for assigning an ASIL. Therefore, an end-to-end run of this methodology takes about 20 minutes.

The FI process and the CARLA simulation took place on a 32-core processor running at 2.99 GHz with 24 GB of memory and a graphics card with 11 GB of memory. Only the RL part is multi-threaded. The contract verification process took place on a 12-core Intel i7-8850H CPU at 4.3 GHz with 16 GB of memory.

5 Discussion

Approach Benefits. The approach of the SAHARA methodology (Section 3) is to ease the effort required by the safety engineer in performing a safety assessment of the component. This process can (semi-)automatically produce simulations and visualisations of component faults.

Thus our addition of ML-based FI is a natural step, as it attempts to optimise the parameters of injected faults such that a more hazardous situation develops (Section 3.3). As in previous work [17], this approach is superior to random-based FI in exploring the fault space in terms of fault coverage and number of simulation to find the first critical fault. The addition of this step may thus allow the safety engineer to discover previously unknown situations where unsafe behaviour occurs and increase fault coverage. Therefore, the simulations and visualisations increase the safety engineer's comprehension of the possible component faults and offer concrete discussion points and insights.

A substantial value of this automatic SAHARA methodology with ML-based FI is the (semi-)automation, leading to new visualisations and classification results produced in a matter of minutes (Section 4), although a computationally powerful machine is required. We envision an assessment workflow where visualisations could be interactively produced during a safety assessment discussion about various scenarios or faults.

Human-in-the-Loop Necessary. At first glance, not having a fully automated framework and keeping a human-in-the-loop is a limitation for SAHARA. However, this is not possible in safety assessment, requiring a tremendous amount of experience and in-depth domain knowledge that is unlikely to be adequately captured by an automaton. The safety engineer's role cannot be replaced entirely, despite the cost of safety assessment discussions. This is due to barriers, including legal responsibilities or insufficient simulation fidelity.

Approach Limitations. Our approach inherits the weakness of ML. For example, the time taken to search for a hazardous situation depends on the proper modelling of the vehicle, environment, and the reward function to steer the search [10, 18]. If the representation of the problem or the reward function is insufficient, then a hazardous situation may not be found. The probabilistic nature of ML also means that it cannot be predicted at what time a hazardous situation will be found. Another issue is that this searching process is computationally expensive and can take a significant amount even on a powerful computer. As the vehicle and component models become more realistic and even more complex, this could limit the applicability of this technique.

Here we suggest two challenges not addressed in this work:

a) For any particular scenario, the situation found by the SAHARA methodology may not be the absolute worst-case due to the infinite and granular space of situations possible. For example, a steering fault may or not cause a head-on crash based on a margin of centimetres or less. Instead, this SAHARA methodology aims to present a representative simulation, such that a safety engineer can recognise the inherent danger in this situation.

b) An extended assessment of controllability and severity levels involves more than just vehicle dynamics models [13, 21]. For example, controllability relies on the driver's reaction (and others), which requires modelling of driver's behaviour [23]. Likewise, severity depends on the dynamics of the driver inside the vehicle, such as possible whiplash injuries or interactions with airbags.

Due to these challenges, we restrict our application of the SAHARA methodology to only providing *representative* hazardous scenario visualisations and hazard classification *suggestions* to assist safety assessments for a safety engineer.

6 Related Work

Assessing safety in the automotive domain is an active field of research, especially in assessing autonomous vehicles [22]. For example, the open-source MOBATSim framework combines a sub-microscopic vehicle simulator with reasoning about faults [24]. Faults are injected into the front distance sensor for vehicles in a platooning scenario, with a Monte Carlo approach to find more hazardous situations. Results are presented by the framework to relate the fault parameters with an indication of safety specifications violations. This approach also addressed the design of safety-critical systems, where faults and scenarios are evaluated for two design variants to choose the safer design [26].

Juez *et al.* examine the role of FI in the context of the ISO 26262 standard and its safety assessment process [12]. The SABOTAGE framework is defined, which performs FI on a lateral control vehicle component, simulates the faulty scenario versus a non-faulty (golden) scenario, and then determines the maximum time that a fault can be present in the system.

In this work, we utilise RL to adjust the fault parameters to provoke a more serious situation [18]. Alternatively, the work of Althoff and Lutz [1] adjusts the scenario parameters themselves, such as doubling the initial speed of the ego vehicle in a scenario or arranging the movement of vehicles to block off lanes or increase the danger of overtaking.

Duracz *et al.* explore the use of *rigorous simulations* to assign severity levels in an ISO 26262 safety assessment context [9]. Rigorous simulations operate on explicit dynamics models and produce provably correct bounds for the behaviour. As in our work, Duracz *et al.* base the severity level on the change in the velocity signal upon collision.

Tuncali *et al.* define STL specifications for both system- and component-level to be proved on a simulation [29]. An example is that when an object is *visible* to sensors, the object must be *detected* by the sensors within a specific time frame. An optimisation framework is then employed to find scenarios that falsify the specifications. In contrast, our work performs the optimisation on the FI to search towards the most hazardous situation, and the specifications are only for hazard classification.

Zapridou *et al.* mirror our work by presenting STL verification of properties on an ACC use case using the CARLA simulator [30]. However, our work focuses on the FI portion of determining safety, and also places the intelligent FI within the SAHARA safety assessment process.

7 Conclusion and Future Work

This work has presented the addition of machine learning-based Fault Injection (FI) to the Simulation-Aided Hazard Analysis and Risk Assessment (SAHARA) methodology, as demonstrated on a safety-critical use case of an Adaptive Cruise Control (ACC). Specifically, Reinforcement Learning (RL) explores the parameters of faults injected into the ACC such that a hazardous situation is provoked. This situation is then simulated in the open-source automotive simulator CARLA [8] to produce a visualisation as well as simulation traces for use in indicating the hazard classification level of the situation. Example situations are shown to demonstrate the applicability of our approach, and timing results indicate that this approach is relatively interactive as it takes only around twenty minutes to complete end-to-end, and less than three minutes to produce a new visualisation.

The natural extension of this work is to validate it within an industrial safety assessment process. In particular, performing a study following the safety engineers as they perform the standard hazard analysis and risk assessment procedure, and then comparing this with our proposed SAHARA with RL-based FI.

Metrics and user surveys would then indicate the time saved and satisfaction with the (semi-)automated approach.

References

1. Althoff, M., Lutz, S.: Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In: 2018 IEEE Intelligent Vehicles Symposium (IV). pp. 1326–1333. IEEE (2018)
2. Bae, I., Moon, J., Seo, J.: Toward a comfortable driving experience for a self-driving shuttle bus. Electronics **8**(9), 943 (2019)
3. Benso, A., Prinetto, P.: Fault injection techniques and tools for embedded systems reliability evaluation, vol. 23. Springer Science & Business Media (2003)
4. Bernaerts, M., Oakes, B., Vanherpen, K., Aelvoet, B., Vangheluwe, H., Denil, J.: Validating industrial requirements with a contract-based approach. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). pp. 18–27. IEEE (2019)
5. Coley, G., Wesley, A., Reed, N., Parry, I.: Driver reaction times to familiar, but unexpected events. TRL Published Project Report (2009)
6. Coppola, R., Morisio, M.: Connected car: Technologies, issues, future trends. ACM Comput. Surv. **49**(3) (Oct 2016)
7. Denil, J., Mosterman, P.J., Vangheluwe, H.: Rule-based model transformation for, and in Simulink. In: Proceedings of the Symposium on Theory of Modeling & Simulation-DEVS Integrative. pp. 1–8 (2014)
8. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
9. Duracz, A., Aljarbouh, A., Bartha, F.A., Masood, J., Philippsen, R., Eriksson, H., Duracz, J., Xu, F., Zeng, Y., Grante, C.: Advanced hazard analysis and risk assessment in the iso 26262 functional safety standard using rigorous simulation. In: Cyber Physical Systems. Model-Based Design, pp. 108–126. Springer (2019)
10. Hauer, F., Pretschner, A., Holzmüller, B.: Fitness functions for testing automated and autonomous driving systems. In: International Conference on Computer Safety, Reliability, and Security. pp. 69–84. Springer (2019)
11. International Organization for Standardization: ISO 26262: Road vehicles-functional safety (2011)
12. Juez, G., Amparan, E., Lattarulo, R., Rastelli, J.P., Ruiz, A., Espinoza, H.: Safety assessment of automated vehicle functions by simulation-based fault injection. In: 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES). pp. 214–219. IEEE (2017)
13. Kemmann, S.: SAHARA-A Structured Approach for Hazard Analysis and Risk Assessments. Ph.D. thesis, Fraunhofer-Institut für Experimentelles Software Engineering (2015)
14. Macher, G., Sporer, H., Berlach, R., Armengaud, E., Kreiner, C.: SAHARA: a security-aware hazard and risk analysis method. In: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 621–624. IEEE (2015)
15. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004)

16. Meyers, B., Gadeyne, K., Oakes, B.J., Bernaerts, M., Vangheluwe, H., Denil, J.: A model-driven engineering framework to support the functional safety process. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). pp. 619–623 (Sep 2019)
17. Moradi, M., Oakes, B., Denil, J.: Machine learning-assisted fault injection. In: 39th International Conference on Computer Safety, reliability and Security (SAFECOMP), Position Paper, Lisbon, Portugal (2020)
18. Moradi, M., Oakes, B.J., Saraoglu, M., Morozov, A., Janschek, K., Denil, J.: Exploring fault parameter space using reinforcement learning-based fault injection. In: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W). pp. 102–109. IEEE (2020)
19. Moradi, M., Van Acker, B., Vanherpen, K., Denil, J.: Model-implemented hybrid fault injection for Simulink (tool demonstrations). In: Chamberlain, R., Taha, W., Törngren, M. (eds.) Cyber Physical Systems. Model-Based Design. pp. 71–90. Springer International Publishing, Cham (2019)
20. Polydoros, A.S., Nalpantidis, L.: Survey of model-based reinforcement learning: Applications on robotics. Journal of Intelligent & Robotic Systems **86**(2), 153–173 (2017)
21. Rafael, A.B.J., Bachir, Z.: Sahara: Simulation aided hazard analysis and risk assessment methodology. Risk Analysis XII **129**, 41 (2020)
22. Riedmaier, S., Ponn, T., Ludwig, D., Schick, B., Diermeyer, F.: Survey on scenario-based safety assessment of automated vehicles. IEEE Access **8**, 87456–87477 (2020)
23. Salvucci, D.D.: Modeling driver behavior in a cognitive architecture. Human factors **48**(2), 362–380 (2006)
24. Saraoglu, M., Morozov, A., Janschek, K.: Mobatsim: Model-based autonomous traffic simulation framework for fault-error-failure chain analysis. IFAC-PapersOnLine **52**(8), 239–244 (2019)
25. Saraoğlu, M., Morozov, A., Söylemez, M.T., Janschek, K.: ErrorSim: A tool for error propagation analysis of Simulink models. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 245–254. Springer International Publishing, Cham (2017)
26. Saraoğlu, M., Shi, Q., Morozov, A., Janschek, K.: Virtual validation of autonomous vehicle safety through simulation-based testing. In: Bargende, M., Reuss, H.C., Wagner, A. (eds.) 20. Internationales Stuttgarter Symposium. pp. 419–434. Springer Fachmedien Wiesbaden, Wiesbaden (2020)
27. Shanahan, D.F.: Human tolerance and crash survivability. Pathological aspects and associate biodynamics in aircraft accident investigation (2004)
28. Singh, K.B., Taheri, S.: Estimation of tire–road friction coefficient and its application in chassis control systems. Systems Science & Control Engineering **3**(1), 39–61 (2015)
29. Tuncali, C.E., Fainekos, G., Prokhorov, D., Ito, H., Kapinski, J.: Requirements-driven test generation for autonomous vehicles with machine learning components. IEEE Transactions on Intelligent Vehicles **5**(2), 265–280 (2019)
30. Zapridou, E., Bartocci, E., Katsaros, P.: Runtime verification of autonomous driving systems in carla. In: International Conference on Runtime Verification. pp. 172–183. Springer (2020)