A Novel Scenario-Based Testing Approach for Cooperative-Automated Driving Systems

Xizhe Zhang, Yuen Kwan Mo, Emil Chodowiec, Yun Tang,
Matthew Higgins, Siddartha Khastgir, Paul Jennings
WMG, University of Warwick, Coventry, United Kingdom, CV4 7AL
{jason.zhang, tony.mo, emil.chodowiec, yun.tang, m.higgins, s.khastgir.1, paul.jennings}@warwick.ac.uk

***Abstract*—This paper presents a scenario-based safety assurance process for Automated Driving Systems (ADSs) combined with the Vehicle-to-Everything (V2X) connectivity aspect. In addition, a novel approach to V2X modelling is introduced and implemented to obtain the required configuration of the V2X parameters for individual system, such modelling approach ensures that the V2X is effective within a system during testing by using a distance-based V2X parameter that correlates to the system speed. Such parameter is then used as the configuration of the V2X model when carrying out the ADS and V2X combined test. Using a pedestrian crossing scenario, a reduction in failure cases is demonstrated by combining the V2X and the ADS. Therefore, a synchronised approach of vehicle, sensor and communication sub-system can improve the overall safety function of the system.**

***Index Terms*—V2X, Connectivity, Scenario-based testing, ADS, Safety**

I. Introduction

Safety Assurance Recent development in Cyber-physical Systems (CPSs) as part of Intelligent Transportation Systems is driven by its benefits. However, the safety certification of CPSs faces many challenges [1]. For Automated Driving Systems (ADSs) and Advanced Driver Assistance Systems (ADASs), scenario-based testing has been widely proposed [1].

A scenario contains the temporal development between several scenes, where a scene is an instantaneous snapshot of the elements within a scenario [2]. The scenario covers the elements defined within the Operational Design Domain (ODD) and their behaviors [3]. ODD refers to the operating conditions under which a system is designed to safely operate, which contains the scenery, environmental conditions, and dynamic elements (e.g., agent types, macroscopic traffic behaviour, system-under-test (SUT)'s speed limit) [4], whereas the behavioural aspects are associated with activities and manoeuvres scenario agents can perform. Underpinned by the ODD, scenario-based testing defines the relevant scenarios, executes them in a valid environment (virtual, physical, hybrid) together with the SUT, collects safety evidence, and outputs testing outcomes [1]. To sum up, a scenario-based testing workflow contains the ***scenarios***, the ***environment***, and the ***safety argumentation***.

Connectivity Although an ADS can function without V2X capabilities, its sensing capabilities of the surroundings (range, precision, and accuracy) are limited by its function, its locations, and its point of view relative to other road users. Cooperative information sharing among ADSs not only helps avoid the contentions of trying to out-predict each other's behaviours, but also enhances road capacity and fuel efficiency. Non-ADS system has already implemented some rudimentary cooperative communications such as turn signals, horn, and brake lights. Front headlights and driver hand gestures are often used to communicate the right-of-way with other road users. Advanced communications for human drivers are however limited because they can be distracting and misleading. ADSs, on the other hand, can be designed to communicate within the same protocol without ambiguity.

Objectives In this paper, a scenario-based and simulation-based testing methodology and implementation involving both an ADS and the V2X protocol are presented. The main contributions include: 1) illustrating a scenario and simulation-based testing of ADS with V2X capabilities, 2) providing evidence on the impact of V2X for ADS safety and performance, 3) introducing a novel connectivity modelling approach, and a V2X-related critical distance parameter to ensure the validity of V2X testing. Considering both the connectivity elements and an ADS, this paper aims to deliver the following objectives:

- **Obj 1** - illustrate a scenario-based and simulation-based testing workflow, as well as the representation of connectivity in scenarios and within simulation execution.
- **Obj 2** - present a novel approach to connectivity modelling, and its incorporation within scenario-based testing.
- **Obj 3** - using example use case to provide a testing baseline for the open-source ADS stack Autoware [5].
- **Obj 4** - determine the connectivity range required for V2X & ADS combined testing, and identify a robust V2X configuration for the system testing.
- **Obj 5** - use the connectivity representation and the ADS together to perform scenario-based testing.

II. Related Work

A. Scenario-Based Testing Workflow

A scenario-based assurance workflow was discussed in [1] and three key components exist at a high level: *Scenario*, *Environment*, and *Safety Evidence* (Figure 1). *Scenario* further covers *Create*, *Format*, and *Store*, which corresponds to the creation of the scenario content, the representation of the created content into description formats, and the storage into

a scenario database such as the Safety Pool™ Scenario Database [6]. *Plan* and *Execute* then determine an execution environment (virtual, physical, or hybrid) and collect execution data. *Analyse* analyses the collected data and checks against pass/fail criteria, such results will then be fed into an optimisation module to create more challenging concrete scenario parameters for the SUT. At the final step, the *Decide* will provide the outcome of the whole process.

B. Scenario

Several scenario description languages [7]–[10] are available for describing scenarios for ADSs/ADASs, covering different abstraction levels. Functional, logical, and concrete scenario abstraction levels were first proposed [11]. A recent paper further adds the abstract scenario level in between functional scenario and logical scenario [12]. To create a coherent and traceable scenario-based development and testing process, as shown in Figure 2 [8], there is a need to establish linkage across different levels [1], [7]. The SDL level 1 from the two-level SDL [7] uses a natural language-based format at abstraction scenario level, the SDL level 2 sits at the logical and concrete scenario level. [8] also documents a texture format covering various abstraction level. [7] and [8] further contributed to the BSI Flex 1889 [13] on a structured natural language based scenario description format. The ASAM OpenSCENARIO [9] and OpenDRIVE [10] are widely used for simulation execution at the concrete scenario level. [14] presented an automated translation workflow, together with its open-source plan, to convert from the two-level SDL into OpenSCENARIO 1.1 and OpenDRIVE 1.6 formats.

Although various scenario description formats exist, the consideration of connectivity within the languages is still limited. At the time of writing, only [15] has embedded the connectivity aspects within the usual driving scenario context. The connectivity features were divided into the road structure aspect within the scenery description (e.g., a communication box), the dynamic aspect (e.g., casting certain messages), and the environmental condition aspect (e.g., the overall coverage of the connectivity).

C. Connectivity

Connectivity has always been a sub-system in safety-related applications. Therefore, the tolerance for failure is far greater than those that include a human in the loop [16]. However, that does not mean various safety protocols do not exist. Safety protocols such as CANBUS [17], MODBUS [18], Time-Trigger Ethernet [19], and EtherCat [20] are designed for machine-to-machine communication in a human in the loop safety system [21]. Key characteristics of these protocols include: messages having a fixed data size [22], communication using Token Ring architecture where there is a master controller polling between each sub-system and the sub-system takes a turn to communicate. A time-critical infrastructure is kept to maintain a watchdog that ensures all stages of the protocol return to their initial conditions. The programming language on communication is often written in Safety C [23], where only one infinite loop can exist in the whole system, memory access is direct only (dynamic addressing is not allowed); There are two processors running in locked steps (processors check each other's execution) and decisions are made in a "three voters, two pass" system. Typically, these systems use communication medium that has a 10^{-9} error probably which is acceptable in the system critical environment [24].

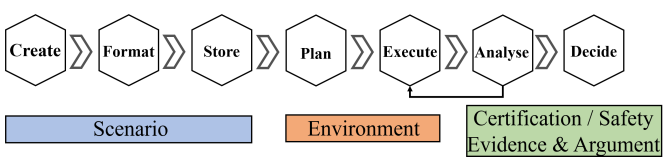

Fig. 1. Scenario-based evaluation continuum with its sub components [1]

D. Virtual Test Environment and System Under Test

Several environment simulation tools are available to provide the virtual 'world' within which driving functions can operate. Among the available simulation tools, they vary in fidelity level and focus on different testing objectives. From the fidelity perspective, they range from visually compelling photo-realistic physics engine-based simulation, to minimalistic graphical rendering of the environment, to purely text-based simulation. From a testing objective perspective, they range from microscopic traffic-based simulation to individual vehicle and pedestrian level simulation, to system level simulation (e.g., planning module simulation), and sensor level simulation [25].

For the SUT, Autoware [5] and Apollo [26] are two of the popular choices for research as well as industrial use cases, they both include a fully autonomous driving stack. On the other hand, rather than incorporating the established AV stacks, one can also implement naive rule-based driving functions for rapid prototyping and development interfacing with different simulation tools.

E. Optimisation Engine

During a scenario-based testing workflow, the parameter spaces defined in a logical scenario format (i.e., parameter value ranges) are explored to identify specific concrete parameter values that induce SUT failures. Various methods have

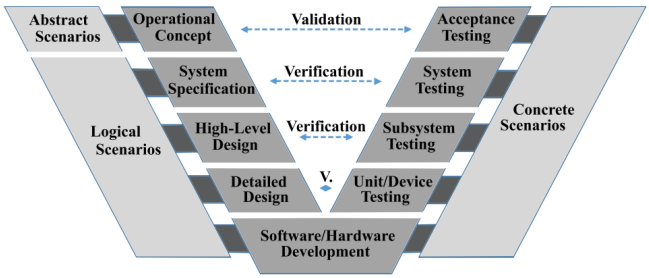

Fig. 2. Scenario abstraction levels mapped to the V model for system development and testing [8]

been published addressing how to effectively derive such failure cases, as the parameter spaces become large, the possible combinations of parameter values will also increase exponentially. One of the basic methods is 'brute force', or grid search over a multidimensional grid; however, such a method can easily reach the resource constraint. Another alternative is to use constraint randomisation, which generates concrete values randomly within the defined value ranges. Such a method was used previously for scenario generation process [27], however for exploring the scenario parameter space during the testing phase it might result into missing key failure cases. [28] demonstrated the use of Bayesian optimisation capabilities to find the edge cases within a scenario parameter space, the optimisation engine learns from the previous execution result and can effectively identify the parameter combinations that violate the pre-defined safety goals.

III. Methodology

A. Connectivity

In order to purely test the connectivity, it is necessary to ensure the input data (e.g., pedestrian location) to the SUT originates from the connectivity, rather than other sensing functions. Within a scenario, a SUT can accumulate its travel distance (d) through a variety of velocities (v_n) in different time sequences (t_n), $d_{total} = \sum_{n=1}^{\infty} v_n t_n$. On the other hand, the V2X protocols operate with time domain only (t_n) across different sub-stages, which may include the starting stage to load the V2X Communication Protocol per instruction in t_1, to operate the instruction of the protocol in t_2, the physical transmission of the protocol as per instruction in t_3, the environment reaction to the transmission in t_4. Similarly, listening to the protocol involves the sampling of the environment in t_5, the translation of samples to data t_6, the interpretation of the data to information in t_7, the action (car manoeuvring) in t_8. As the SUT can adjust its velocity at any time during the test, and different systems can achieve different velocity in the same period of time, a pure V2X test on a SUT would need a dynamically changing distance to compensate for the vehicle's potential actions, and ensure that the groundtruth data feed lies in the V2X communication rather than other sensing layers. As shown in Figure 3, within the same time period from the SUT perspective, if the total time for the communication multiply by the SUT speed ($d_{success}$) is less than the SUT's actual distance of travel (d_{total}), then the destination groundtruth data will be received by the SUT from the V2X earlier than its own sensing (providing in this case SUT only knows groundtruth at its current location), hence the V2X is being tested. On the other hand, if the communication time multiply by the SUT speed (d_{fail}) exceeds the actual travel distance by the SUT (d_{total}), then the destination groundtruth received by the SUT will come from its own sensing rather than V2X. In this paper, a novel methodology is introduced to quantify the communication requirements from a time based approach into a distance based approach, $\lim_{d\to\infty} f(d_n) = f(t_n)$ where $f(d_n)$ is the distance the SUT will travel within the communication time, and $f(t_n)$ is the communication time. The distance SUT will travel for successful communication to be made $d_{success}$ is therefore the sum of distances travelled by the SUT at individual communication stages, $d_{success} = \sum_{n=1}^{\infty} d_n$.

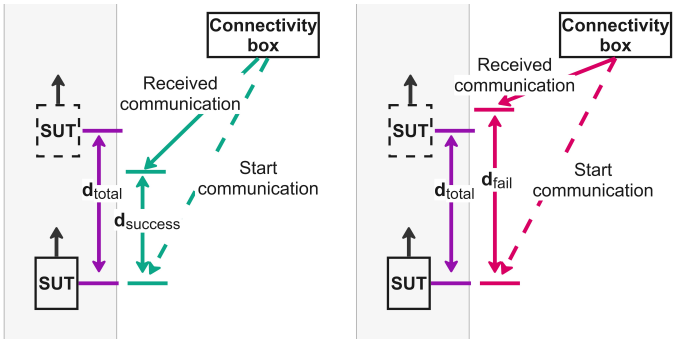

Fig. 3. Pass and failed of the V2I Communication

Paradoxically, each of the communication protocol stages could pass or fail, and any stage that fails would result in the delay or breakdown of the V2X communication protocol as conjunction, $S_{overall} = S_{Stage_1} \wedge S_{Stage_2} \wedge S_{Stage_3} ... S_{Stage_n}$ where S represent the success, the system might retry the failed stage until succeed and then move to the next stage. The distance required to successfully complete all the stages of the protocol must be less than the total distance in the test $d_{success} \leq d_{total}$ to pass. Furthermore, distance related to individual stages d_n also needs to be much less than the total distance $d_n \ll d_{success} \leq d_{total}$. An example of a V2X protocol could contain an acceptable tolerable failure if the number of stages in the protocol is low as seen in Figure 3. $d_{total} \geq d_{tolerance} + \sum_n d_n$, where the acceptable failure in distance $d_{tolerance}$ is effectively the total distance d_{total} subtracted from the number of stages n_{stage} in protocol and each success distance d_{total}.

B. Scenario Format and Conversion

As mentioned in Section II-B, the V2X elements have been incorporated into the two-level abstraction scenario formats [7], [15] which will cover the abstract scenario level using a natural language format, and the logical scenario level using parameter ranges. Through the conversion [14] to ASAM OpenX formats [9], [10] for simulation execution, the concrete scenario level can also be covered, Figure 4 illustrates such a process. Currently, such a conversion process covers the non-V2X elements.

In SDL level 1 and level 2, the representation of connectivity spreads across the scenery elements (for describing the communication devices), the dynamic elements (for describing the communication activities), and the environmental elements (for defining the connectivity coverage). However, such direct representation of connectivity in the ASAM OpenX format is not yet supported, especially for the dynamic and environmental aspects. In this paper, the scenery aspect of the connectivity (i.e., the communication device) is translated as scenario objects with their positions defined in the ASAM OpenDRIVE file. For the dynamic and environmental aspects,

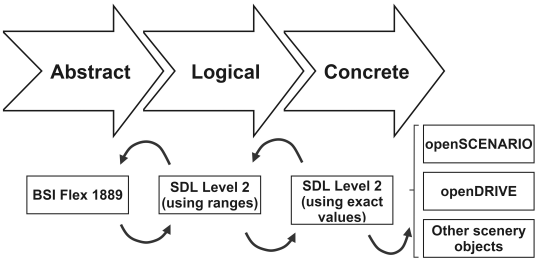

Fig. 4. Conversion between different scenario abstraction levels [29]

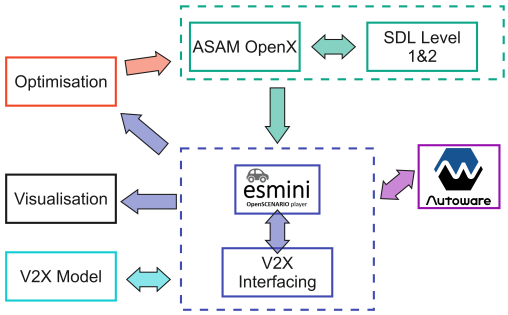

Fig. 5. Virtual execution framework with its components

they are implemented by an external Python script that parses directly the SDL connectivity descriptions and incorporates the receiving/sending of certain ground truth data between scenario objects and actors based on the scenario. The connectivity model (in Section III-A) is also incorporated with the script.

C. Execution Framework

The execution framework utilises a modularised approach, it broadly consists of a *scenario conversion*, a *simulation core (with V2X interfacing)*, an *ADS*, an *optimisation engine*, a *V2X model*, and optionally a *visualisation engine*, as seen in Figure 5.

It can be seen that the whole workflow starts with the SDL level 1&2 descriptions of an intended scenario. Such a scenario is then converted to the ASAM OpenSCNEARIO 1.1 format and OpenDRIVE 1.6 format. The open source simulator esmini [30] has been chosen as the simulation engine for this implementation as it enjoys a high degree of ASAM OpenX support and contains a detailed user guide for integration. An additional V2X interface module handles the communication with the V2X model for calculating the data transmission properties, it also synchronises with the SDL description to perform the communications of certain ground truth data with the SUT. Please note that the V2X model can be used for: 1) identifying V2X requirements, and 2) testing a specific V2X parameter combination. An example of the first category can be that given a desired time or distance advantage as the input, the valid V2X parameter combinations will be calculated and output. An example of the second category can be that given a set of V2X parameters, the model will calculate the corresponding communication time or distance and use it for the scenario-based testing process to explore the SUT failure modes. For the SUT, in this implementation, the open-source Autoware [5] AV stack is chosen. This implementation considers the sensing and perception layers only at the object-list level, therefore the focus is more on the planning and control layers. An optimisation engine is then integrated to analyse the execution data against pre-defined scenario criteria, the outcome of the optimisation engine is a new set of concrete scenario parameters which are used to create a new OpenSCENARIO file. In this implementation, the Bayesian Optimisation algorithm is used as the optimisation engine. Finally, an additional visualisation module is also integrated within the execution framework to provide photo-realistic replays of the scenario executions. Such visualisation module in the current implementation is only used for human consumption since the SUT is running directly on the object list, however, it can also be used to provide raw sensor input to the ADS for broader system-level testing.

D. Optimisation Objectives

Bayesian Optimisation is a class of machine-learning-based optimisation methods for optimising objective functions that take a long time to evaluate. At its core lies the Gaussian Process (GP) models which derive a prior over the black box function that is being optimised [28]. It is particularly useful in situations where traditional optimisation methods may be too slow or too expensive to use. Bayesian optimisation builds a probabilistic model of the objective function, using Bayesian inference to iteratively update the model as new observations are made. The model is used to make informed decisions about where to sample the objective function next, in order to find the optimum with the least amount of function evaluations. In this study, a simple optimisation objective of *collision* between the SUT and other scenario actors is used, and the scenario parameters related to the SUT and other actors are used as the input to the optimisation engine. The *collision* can be treated as the occurrence of any overlapping between the bounding boxes that define the SUT and other actors, and the scenario parameters include SUT/other actors' speed, position, etc.

E. Three stages of the investigation

To meet the objectives set out in the **Introduction**, and to demonstrate the capabilities of such workflow combining the V2X feature, this study will be divided into three stages.

Stage I will focus on a non-V2X scenario, and carry out a scenario-based testing process on Autoware using the Bayesian optimisation to identify failure conditions.

Stage II will then investigate providing a distance-based advantage parameter derived from the connectivity range, and investigate a suitable V2X parameter combination for the SUT. Such distance advantage will be analysed and suggested from the V2X model considering various data transmission stages within the communication.

Stage III will utilise the V2X-based distance parameter from *Stage II*, incorporate it using a V2X sensor modelled within the simulation to supply ground truth data to the SUT. The testing process over the same scenario and SUT (Autoware) will be performed with the V2X sensor integrated. Comparison will be made between the non-V2X ADS and V2X incorporated ADS.

IV. Case Study

A. Problem Statement

Since this study is not focused on scenario generation, an existing scenario from the Safety Pool™ Scenario Database [6] is used for the case study. The Safety Pool™ Scenario Database contains more than 250,000 scenarios, generated from over eight different scenario generation methods covering both data-based and knowledge-based approaches, it is the world's largest public database at the time of writing. Across different data sources and scenario generation methods, the *pedestrian crossing scenario* has been identified. A variation of the scenario is covered in the UN regulation R157 on ALKS (Automated Lane Keeping Systems) [31], also in the ISO 22737 on LSAD (Low-Speed Automated Driving) test procedures [32], as well as in the EURO NCAP Car-to-Pedestrian scenarios [33]. A schematic can be seen in Figure 7.

Based on the pedestrian crossing scenario, the following steps are taken: 1) identify the parameter combinations of V_{Ped}, V_{Car}, d that result in a pedestrian-SUT collision, 2) introduce a distance-based advantage parameter derived from the V2X model, 3) identify failure parameter combinations of the same SUT but with V2X incorporated in the test, and benchmark failure combinations found in step 1.

B. Scenario representation

The scenario description at the abstract scenario level is described using the BSI Flex 1889 with a structured natural language format [13], which is presented in Figure 6. The process of scenario development follows the conversion analogy presented in Figure 4 in order to derive logical and concrete-level scenarios while maintaining traceability. Based on Figure 6, a logical-level scenario description using the WMG SDL level 2 format is created to provide more details and use parameter ranges. Finally, the concrete scenario is generated to provide the exact scenery and chain of events with concrete parameters using the translation toolchain detailed in [34]. At this level of abstraction, the ASAM OpenSCENARIO 1.1 and OpenDRIVE 1.6 formats are used.

The scenario takes place between 03:00 to 06:00, under a lighting condition of 100.0 to 25000.0 'lx' , and a cloud condition of 0 to 1 'oktas'. With cellular communication. There is no junction present.

There is 1 road, Road R1. Road R1 is a 175 to 225 'm' straight B road, a level plane. There are 2 lanes on Road R1, Lane L1 and Lane L2. The travel direction between Lane L1 and Lane L2 is the opposite. Lane L1 has broken line, Lane L2 has broken line. There is vegetation on Road R1 ,there are streetlights on Road R1 and there is a communication unit with a casting type of multicast as Structure CB1.

There is 1 vehicle and 1 pedestrian, vehicle ego and pedestrian PD1. Vehicle ego is at Road R1 and Lane L2. Pedestrian PD1 is at Road R1 and Lane L2, and pedestrian PD1 is at front right of vehicle ego with a distance of 40.1 to 120.1 'm'. When vehicle ego is driving, pedestrian PD1 walks across vehicle ego at its front right with a distance, at a very slow speed of 1.12 to 6.71 'mph'.

Fig. 6. Scenario description in BSI Flex 1889 format [13]

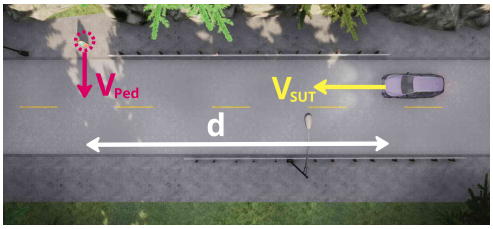

Fig. 7. Diagram of the pedestrian crossing scenario

C. Stage 1 - ADS Only Testing

Similar to Figure 6 (without the connectivity aspect), the scenario in Stage 1 involves two actors: a SUT equipped with an ADS (Autoware) and a pedestrian intending to cross the road. A visual representation of this scenario is illustrated in Figure 7. The intended safe behaviour for Stage 1 is that the vehicle stops or avoids collision with the pedestrian. This step investigates three key scenario parameters: the speed of the SUT (V_{SUT}), the speed of the pedestrian (V_{Ped}), and the vehicle's initial distance to the pedestrian (d). The Bayesian Optimisation (BO) algorithm, described in Section II-E, explores the parameter values to identify potential collision events. To restrict optimisation engine exploration space, bounds were determined for each of the parameters as shown in Table I.

The testing objective is to determine the minimum distance between the pedestrian and the SUT, thus to determine non-collision and collision cases. During each execution of the logical scenario described in WMG SDL level 2, the BO engine conducted 20 iterations with the objective to drive collision

TABLE I
Bayesian Optimisation Parameters Bounds

Parameter	Unit	Value
Initial distance [d]	m	40 to 120
SUT initial speed [V_{SUT}]	m/s	5 to 35
Pedestrian speed [V_{Ped}]	m/s	0.5 to 3

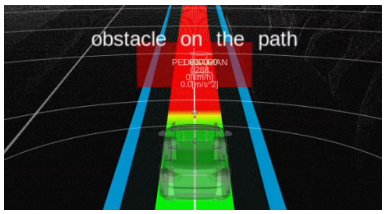

Fig. 8. Example of the object detection by Autoware

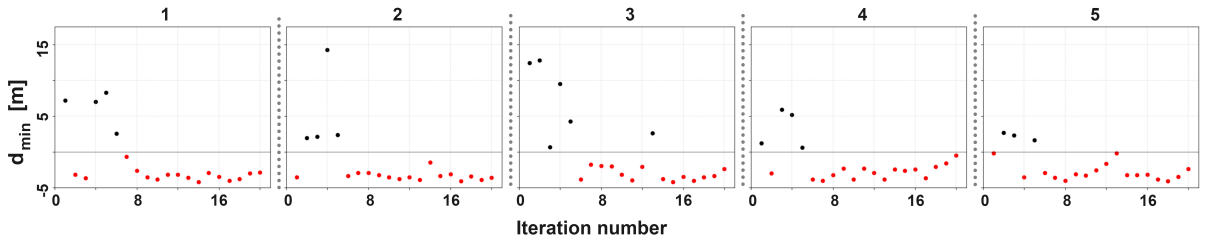

Fig. 9. Results of Stage 1, non-V2X testing

events as described in III-D. The same logical scenario was executed five times to demonstrate a wider range of results. The Autoware platform is utilised to represent the SUT, Figure 8 illustrates a visualisation of the pedestrian detection made by Autoware. As can be seen that the SUT detects the obstacle on the trajectory path and subsequently triggers a chain of events internally.

The results from the BO-identified parameters and the scenario execution are shown in Figure 9. Each dot represents an execution of the OpenSCENARIO and OpenDRIVE pair. The y-axis of the graph represents the minimum distance attained between the ego vehicle and the pedestrian within each execution, while the x-axis indicates the iteration number. Furthermore, the non-collision cases are denoted by black points, while the collision cases are represented by red points. These instances result in negative values, indicating that the bounding boxes of entities overlapped. Initially, it is evident that in the majority of scenarios, a collision occurred. The achieved distance exhibits variability during the initial iterations, as the BO engine is exploring a broad parameter space. Once the engine identifies the failure cases, it narrows its exploring area to the proximity of the collision boundary, enabling the identification of collision-prone scenarios. All five executions show a consistent trend, indicating that failure cases are detected at approximately iteration number six. Based on the data, collisions occurred in 79% of the scenarios. It demonstrates that BO is effective in identifying the scenario parameters that cause the ADS to fail. Please note, this paper is not intended to investigate the capabilities of the optimisation algorithm, but rather to demonstrate a complete testing workflow using V2X in later stages, therefore the BO is used as an example.

D. Stage 2 - Connectivity

There are two main objectives in Stage 2: 1) **identify a V2X range** for the Stage 3 - V2X and ADS testing using the pedestrian scenario, 2) introduce a robust **V2X model** and **identify the V2X parameter combination** for the system, such that the system cannot cheat and gain knowledge advantage of a target object using sensing functions other than the V2X to pass the V2X test (i.e., maintaining $d_{success} \leq d_{total}$).

For the V2X range - the SUT would send a V2I protocol to the communication box and the communication box would reply back to the SUT. In comparison to Stage 1, the V2I system is effectively increasing the range of an ADS 'sensing layer' to detect (via communication) its surrounding. Similar to an ADS extends its range of sensing by the use of road signs and traffic lights. An ADS that communicates with the road system can cooperatively optimise the road capacity to support other systems by communicating its velocity, its separation distance with other road users, and the road capacity to enable the re-routing of other road users to alternative paths. A vehicle-to-vehicle/pedestrian communication network (V2I) is only limited by its radio reach to the connection box. In this example, a V2I system is being proposed and the communication distance starts from 100m to 200m [35] away from the ADS system to the connection box. The formula for calculate the distance is given by, $D_{total} = \lambda/4\pi\sqrt{P_t D_r D_t/(P_r)}$ [36], where P_t and P_r are the radio power between receiver and transmitter respectively and D_r and D_t, λ is the wavelength in meters (0.125m for 2.4GHz). As long as the received transmission power remains high $P_r/P_t > (1/4\pi D_{total})^2$ and both transmitter and receiver antennas have a $D_t D_r$ radio coefficient close to 1, power loss in free space is equal to $1/4\pi$ square relative to distance. The modern receiver has a sensitivity of -70dBm and so 100m to 200m distance is achievable in our simulation.

On the other hand, for V2X protocol only testing - since there are 8 stages in our protocol to transmit and receive a reply message. The breakdown of 8 stages of the protocol is simulated as $d_{success} = d_1 + d_2 + d_3, ... + d_8$, each stage is tested by assigning an arbitrary distance to complete with the sum of the distances equal to $d_{success}$. If the ego vehicle pre-emptively reduces its potential velocity to cheat the V2I test at Stage 2, the relative distances d_n of individual stages would be shorter to ensure $d_{success} \leq d_{total}$ for each test. For example, a SUT travelling at 100 mph the $d_{success}$ will be different compared with if it is travelling at 10 mph. At 10 mph, the d_{total} will be much smaller for the same time duration, therefore the corresponding $d_{success}$ will be smaller. Imagine, if a system designed for driving at 100 mph deliberately drives at 10 mph, the $d_{success}$ will be small from a static V2X model. Under such $d_{success}$, if the system then travels at 100 mph for connectivity testing, it will have already gained destination information before V2X communicates to it, and act accordingly to pass, but the V2X model of $d_{success}$ will still assume it is traveling at 10 mph and therefore consider it as a pass. To prevent such 'cheats' to happen by the system,

TABLE II
d_n AND PASS/FAIL AT EACH PROTOCOL STAGE

Protocol Stages	Distance	Value
Loading of protocol instruction [d_1]	1m	pass
Hardware protocol instruction [d_2]	3m	pass
RF physical transmission [d_3]	1m	pass
Environment reaction [d_4]	1m	fail
RF physical transmission [d_3]	1m	pass
Environment reaction [d_4]	1m	pass
Sampling (Listening) [d_5]	1m	pass
Sampling translation [d_6]	1m	pass
Interpret data to info [d_7]	3m	pass
Car manoeuvring [d_8]	8m	pass
Total [$d_{success}$]	21m	pass

the calculation of $d_{success}$ will need to be dynamically adapted to the system and consider the ODD of the SUT where the maximum designed safe operating speed is specified. In Stage 2, the SUT is tested multiple times and the V2X parameter combinations generated take into consideration the system. This is done by simply operating the SUT to drive at different levels of capacity and calculate the d_{total} (as shown in Figure 3), such d_{total} will be used as the maximum $d_{success}$ for the V2X modelling, and consequently, the parameter combinations can be determined.

Stage 2 will use $d_{success}$ to measure the responsiveness of the V2I protocol by testing its design/coding/performance in relative time according to distance. This will then encourage vehicle designers to increase their resilience, reliability, and reciprocity in an effective exchange of data for SUT with higher velocity potential. The ego vehicle may fail each stage of the protocol, and each stage could only advance until they have passed the previous stages, this is handled by the processor and watchdog inside the ADS V2I system as explained in the related works. Since in this scenario, the road system can extend indefinitely to work out its $d_{success}$, it is possible for all ADS to pass V2I eventually even when $d_{success} \to \infty$. A typical V2V SUT testing model may include but is not limited to the ranges illustrated in Table II.

E. Stage 3 - ADS with V2X Combined Testing

In Stage 3, the V2X capability is incorporated to enable data transmission to the vehicle from added V2X sensor. This is accomplished by providing the Autoware system with ground truth data based on the suggested range in Stage 2, in this case, 130m is chosen. Under such a V2X range, the vehicle is aware of its surroundings much earlier than in the ADS-only case (Stage 1). This stage utilises the same scenario design as described in IV-C. A V2X communication box is integrated to firstly monitor and obtain crosswalk data, and secondly transmit the information to the SUT. The scenario setup including the V2X sensor is shown in Figure 10. Such feed of ground truth data provides more time for the vehicle's planning module to analyse the data and determine an appropriate course of action.

The results for the Stage 3 are present in Figure 11. It can be observed that in most iterations collision does not occur.

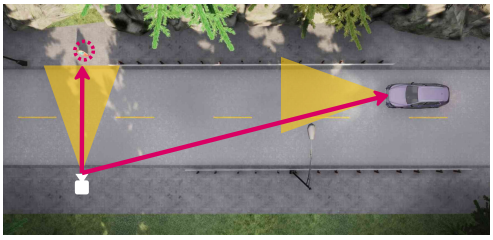

Fig. 10. Schematic diagram of the V2X communication enabled scenario

In execution numbers 2, 4, and 5, no collision is recorded. During the later iterations of execution numbers 1 and 3, collision cases can be observed. The early iterations across all executions present more diverse values of the minimum distance [d_{min}] as the BO explores a wide range of parameters. Subsequently, the values of [d_{min}] decrease after the fifth iteration. Execution number 2 exemplifies this trend. A general observation is that the obtained results for [d_{min}] remain close to zero, with a range of 0m to 5m, indicating no collision cases. Therefore it can be assumed that the SUT has enough time to act and stop before the pedestrian. Only 7% of scenarios failed which is a decrease of 72% in comparison to Stage 1 results. It demonstrates that BO process encountered more difficulties in challenging the system and identifying collision cases than in Stage 1. Therefore, the V2X-enabled scenario demonstrates enhanced safety when compared to the ADS-only systems.

V. CONCLUSION

In this paper, a three-stage investigation for testing ADS is presented. Stage 1 contains an ADS only testing (without connectivity) using the Bayesian Optimisation engine to identify system failure cases. Stage 2 presents: 1) a multi-stage V2X modelling framework to dynamically derive the required V2X parameters based on the SUT speed, 2) introduce a V2X-based distance advantage parameter which can be used in Stage 3. Stage 3 then incorporates this V2X distance parameter within the testing and explores the system failure cases with the consideration of the connectivity aspect. It was observed that the SUT with V2X capability showed increased safety (i.e., less collision cases). Furthermore, the novel and robust V2X modelling technique can dynamically provide the required V2X parameter combinations for any given SUTs. This paper also provides first-hand evidence of incorporating V2X into the ADS testing framework.

ACKNOWLEDGEMENT

The work presented in this paper has been supported by UKRI Future Leaders Fellowship (Grant MR/S035176/1), Department of Transport, UK, and Transport Canada. The authors would like to thank the WMG center of HVM Catapult, and WMG, University of Warwick, UK for providing the necessary infrastructure for conducting this study. No new data were created in this study.

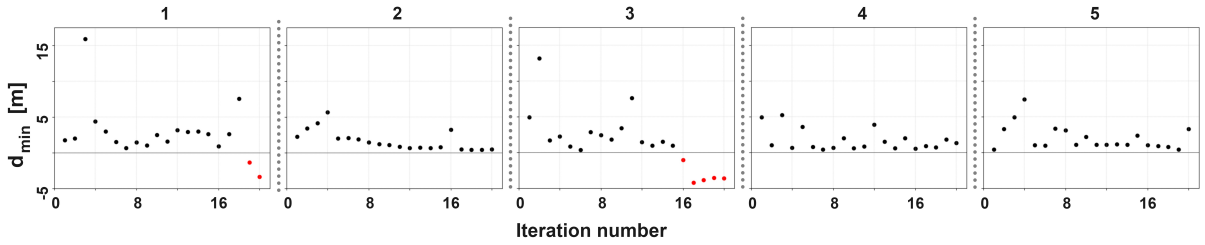

Fig. 11. Results of the V2X enabled testing

REFERENCES

[1] X. Zhang, S. Khastgir, H. Asgari, and P. Jennings, "Test Framework for Automatic Test Case Generation and Execution Aimed at Developing Trustworthy AVs from Both Verifiability and Certifiability Aspects," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 312–319.

[2] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pp. 982–988, 2015.

[3] X. Zhang, S. Khastgir, and P. Jennings, "An ODD-Based Scalable Assurance Framework for Automated Driving Systems," in *WCX SAE World Congress Experience*. SAE International, 2023.

[4] ISO, "ISO/DIS 34503 - Road Vehicles - Test scenarios for automated driving systems - Taxonomy for operational design domain," 2022.

[5] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," *ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pp. 287–296, 2018.

[6] "Safety Pool^{TM} Scenario Database." [Online]. Available: https://www.safetypool.ai/

[7] X. Zhang, S. Khastgir, and P. Jennings, "Scenario Description Language for Automated Driving Systems: A Two Level Abstraction Approach," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020, pp. 973–980.

[8] F. Bock, C. Sippl, A. Heinzz, C. Lauerz, and R. German, "Advantageous usage of textual domain-specific languages for scenario-driven development of automated driving functions," in *2019 IEEE International Systems Conference (SysCon)*, 2019, pp. 1–8.

[9] ASAM, "OpenSCENARIO V1.2.0," 2022. [Online]. Available: https://www.asam.net/standards/detail/openscenario/

[10] ——, "OpenDRIVE V1.7.0," 2021. [Online]. Available: https://www.asam.net/standards/detail/opendrive/

[11] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *IEEE Intelligent Vehicles Symposium*. IEEE, 2018.

[12] C. Neurohr, L. Westhofen, M. Butz, M. Bollmann, U. Eberle, and R. Galbas, "Criticality Analysis for the Verification and Validation of Automated Vehicles," *IEEE Access*, vol. 9, no. i, 2021.

[13] BSI, "BSI Flex 1889 - Natural language description for abstract scenarios for automated driving systems - Specification," 2022.

[14] A. Anastasio Bruto da Costa, P. Irvine, X. Zhang, S. Khastgir, and P. Jennings, "Translating Automated Vehicle Test Scenario Specifications Between Scenario Languages: Learnings and Challenges," in *Proceedings of the Driving Simulation Conference 2022 Europe VR*, 2022, pp. 65–72.

[15] P. Irvine, P. Baker, Y. K. Mo, A. B. Da Costa, X. Zhang, S. Khastgir, and P. Jennings, "Vehicle-to-Everything (V2X) in Scenarios: Extending Scenario Description Language for Connected Vehicle Scenario Descriptions," in *Intelligent Vehicles Symposium (IV)*, 2022, pp. 548–555.

[16] E. Suhir, "Human in the loop: Predicted likelihood of vehicular mission success and safety," *Journal of Aircraft*, vol. 49, no. 1, pp. 29–36, 2012. [Online]. Available: https://doi.org/10.2514/1.C031418

[17] H. Thompson, H. Benitez-Perez, D. Lee, D. Ramos-Hernandez, P. Fleming, and C. Legge, "A canbus-based safety-critical distributed aeroengine control systems architecture demonstrator," *Microprocessors and Microsystems*, vol. 23, no. 6, pp. 345–355, 1999.

[18] L. Xuan and L. Yongzhong, "Research and implementation of modbus tcp security enhancement protocol," *Journal of Physics: Conference Series*, vol. 1213, no. 5, p. 052058, jun 2019.

[19] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (tte) design," pp. 22–33, 2005.

[20] M. Rostan, J. E. Stubbs, and D. Dzilno, "Ethercat enabled advanced control architecture," pp. 39–44, 2010.

[21] I. Stojmenovic, "Machine-to-machine communications with in-network data aggregation, processing, and actuation for large-scale cyber-physical systems," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 122–128, 2014.

[22] Y. K. Mo, M. S. Leeson, and R. J. Green, "Deterministic ethernet using a network traffic oscillator," pp. 573–583, 2015.

[23] J. Devietti, C. Blundell, M. M. K. Martin, and S. Zdancewic, "Hardbound: Architectural support for spatial safety of the c programming language," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 2, p. 103–114, mar 2008.

[24] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A triple core lock-step (tcls) arm® cortex®-r5 processor for safety-critical and ultra-reliable applications," pp. 246–249, 2016.

[25] A. Wallace, S. Khastgir, X. Zhang, S. Brewerton, B. Anctil, P. Burns, D. Charlebois, and P. Jennings, "Validating Simulation Environments for Automated Driving Systems Using 3D Object Comparison Metric," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 860–866.

[26] "Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving." [Online]. Available: https://github.com/ApolloAuto/apollo

[27] S. Khastgir, G. Dhadyalla, S. Birrell, S. Redmond, R. Addinall, and P. Jennings, "Test Scenario Generation for Driving Simulators Using Constrained Randomization Technique," *SAE Technical Papers*, vol. 2017-March, no. March, 2017.

[28] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization," *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, pp. 1961–1967, 2019.

[29] Zenzic.io, "Simulation - Verification and Validation," Tech. Rep., 2022. [Online]. Available: https://zenzic.io/innovation/simulation/

[30] K. e. Al., "Environment Simulator Minimalistic (esmini)."

[31] UNECE, "UN Regulation No. 157 - Automated Lane Keeping Systems (ALKS)," 2021.

[32] ISO, "Intelligent transport systems — Low-Speed Automated Driving (LSAD) Systems for Predefined routes — Performance requirements, system requirements and performance test procedures - ISO 22737," 2021.

[33] NCAP, "Euro NCAP Test Protocol-AEB VRU systems," *November*, no. June, 2020. [Online]. Available: https://cdn.euroncap.com/media/58226/euro-ncap-aeb-vru-test-protocol-v303.pdf

[34] A. Anastasio, P. Irvine, X. Zhang, S. Khastgir, and P. Jennings, "Translating Automated Vehicle Test Scenario Specifications Between Scenario Languages: Learnings and Challenges," in *Proceedings of the Driving Simulation Conference 2022 Europe VR*, 2022, pp. 65–72.

[35] B. Gallagher, H. Akatsuka, and H. Suzuki, "Wireless communications for vehicle safety: Radio link performance and wireless connectivity methods," *IEEE Vehicular Technology Magazine*, vol. 1, no. 4, pp. 4–24, 2006.

[36] H. Friis, "A note on a simple transmission formula," *Proceedings of the IRE*, vol. 34, no. 5, p. 254–256, 1946.