WARWICK
THE UNIVERSITY OF WARWICK

Manuscript version: Author's Accepted Manuscript
The version presented in WRAP is the author's accepted manuscript and may differ from the published version or Version of Record.

Persistent WRAP URL:
http://wrap.warwick.ac.uk/154179

How to cite:
Please refer to published version for the most recent bibliographic citation information.

Copyright and reuse:
The Warwick Research Archive Portal (WRAP) makes this work by researchers of the University of Warwick available open access under the following conditions.

Copyright © and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners. To the extent reasonable and practicable the material made available in WRAP has been checked for eligibility before being made available.

Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge. Provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.

Publisher's statement:
Please refer to the repository item page, publisher's statement section, for further information.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk.

warwick.ac.uk/lib-publications

Test Framework for Automatic Test Case Generation and Execution aimed at developing Trustworthy AVs from both verifiability and certifiability aspects

Xizhe Zhang
WMG, University of Warwick
Coventry, United Kingdom
Jason.Zhang@warwick.ac.uk

Siddartha Khastgir
WMG, University of Warwick
Coventry, United Kingdom
S.Khastgir.1@warwick.ac.uk

Hamid Asgari
Thales UK Research, Technology & Innovation
Reading, United Kingdom
https://orcid.org/0000-0002-9317-7045

Paul Jennings
WMG, University of Warwick
Coventry, United Kingdom
Paul.Jennings@warwick.ac.uk

***Abstract*— Recent developments in the testing and safety assurances of Automated Driving Systems (ADSs) have shifted from traditional distance-based approach (i.e. vehicle miles travelled) to scenario-based approach. Various studies have been conducted on different aspects of the scenario-based testing workflow such as scenario generation, scenario description language, and scenario analysis. This study intends to build on top of the individual functional modules introduced for scenario-based testing and contributes towards a common test framework, based on the experience gained from the Innovate UK's OmniCAV project. The Test Framework introduced in this paper consists of: Description of Test Scenarios, Different Types of Testing and Allocation of Tests, Generation of Test cases, Data Collection, & Analysis, Correlation of obtained Results and Evidence Gathering for functional and behavioral verification. The framework and processes defined here have relevance to real-world practice and contribute towards both verifiability and certifiability of ADSs.**

Keywords—Verification & Validation, Autonomous Vehicles, scenario-based testing, testing framework, scenario and test-case generation, scenario description language

I. INTRODUCTION

Verification and Validation (V&V) are vital parts of the development and deployment of any engineering system. The V&V processes are well established in more mature sectors of engineering such as aerospace and traditional automotive systems. However, they are not as well developed in areas such as autonomy. The V&V ultimately should enable the safe operation of automated vehicles and safety of people. Systems are verified with respect to the specified requirements. Verification methods can be defined to be: 'methods by which confidence can be gained in the correctness of a system with respect to its specification [1]. These methods can be divided into formal verification, dynamic and virtual testing. Formal verification attempts to prove at least some degree of correctness of the system model with respect to requirements; dynamic testing employs test instances to gain confidence in the correctness of the actual system. Development of virtual environment representing the real world (e.g., synthetic environment and models) is of great importance for conducting verification, especially for automated vehicles, as limited physical testing can be performed due to the state space growth. The future state ($\underline{x}_{k+1}$) of Automated Driving Systems (ADSs) will depend not only on the current state ($\underline{x}_k$) and environment ($\underline{u}_k$), but also on the past states and environment of the system from which it may have learnt or adapted, this is expressed as below. f_{aut} is the function of autonomy.

$$\underline{x}_{k+1} = f_{aut}(\underline{x}_k \ldots \underline{x}_1, \underline{u}_k \ldots \underline{u}_1)$$

Due to the large state space, verifying every possible state is likely to be very difficult in practical terms.

In general, this system-level behavior must satisfy the defined functional and operational safety requirements. Appropriate evidence needs to be collected to make sure an ADS behavior has been verified with respect to the requirements. Gathering evidence must show that all the applicable requirements of the rules and regulations have been conformed to. This requires the development of valid test scenarios and scalable techniques for verifying that the requirements for autonomous functions are consistently met and are traceable. Virtual testing also allows scenario play back, proving that there is sufficient confidence that the defined requirements are satisfied by the implementation of each element and system's behavior (i.e., ADS software), and will sufficiently be safe throughout its life-time. Gathering the result allows evaluation of the adequacy of current safety assurance arguments (pass/fail criteria) and is vital for ADS certification. It should be noted that developed capabilities potentially have cross-domain usage. For example, the capabilities developed for ADSs domain can be assessed for adequacy and used in other domains such as Maritime, Rail, Civil Aviation, and Military Aviation.

Traditionally, distance-based metric, i.e. vehicle miles travelled, has been used to demonstrate technology maturity and provide safety assurance for driving systems. However, for ADSs, Kalra et. al suggested that they would need to be driven for 11 billion miles to demonstrate they are 20% better than human drivers [2]. Therefore, for higher levels of automation a distance-based verification approach where driving many test miles on test grounds and public roads is not an economically viable solution. This led to the shift from distance-based approach to a scenario-based approach for ADS safety assurance. Various studies have been published on different aspects of the scenario-based testing workflow such as scenario generation [3][4][5], scenario description format [6][7][8], and scenario analysis [9][10][11]. The auto-generated test cases from ADS scenarios can also be replicated in other domains where specific definition languages and ranged values are used.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

Consideration is required to see which applications can be used directly with little or no change to the system architecture developed for the ADS use-case, and which applications will require further development or changes to the architecture. Other domain use cases including Maritime Autonomous Systems (MAS) are being developed for use across a range of sectors. Relevant methodologies enabling their certification and assurance are also urgently needed. In general, some of the challenges towards V&V of the ADS and other Autonomous Systems (AS) are as follows:

- Development of computationally scalable techniques to formalize and verify the requirements for autonomous functions (for consistency, traceability, and conducting V&V).
- Demonstrate (by gathering evidence) that all the applicable requirements of the rules and regulations have been conformed to.
- Collect the evidence and demonstrate that system-level behavior satisfies the defined functional/ operational cyber security and safety requirements.
- Demonstrate (prove) that there is sufficient confidence that the defined requirements are satisfied by the implementation of each element and system's behavior and will be sufficiently safe throughout its life.
- Evaluate the adequacy of current safety assurance arguments (pass/fail criteria) for certification of AS; How much of a safety case needs updating for a small system change or adding features?
- Address the need for mapping certification requirements to test regimes.
- Devise new or improve the existing test methods, techniques and tools for scalability of testing. Build the means/tools for test decomposition and systematic generation of test scenarios (scenario variation), parameterization of tests, and reproducible concrete test cases.
- Allocation of the tests and use of dynamic testing, virtual testing (scenario play back), or formal verification (in itself or in combination) as the effective means of verifying AS.
- Ascertain in how to secure the synthetic environment itself in order to collect valid results?
- Establish in how the V&V results obtained e.g. from certified simulation can aid to what degree and effect to provide sufficient evidence for assurance and certification purposes?
- Realize that cyber security feature adds complexity to V&V tests. AS test ecosystems should include public, virtual, controlled and cyber-physical testing environments (for cyber-physical tests)
- Determine some metrics to see "verification is actually done".

The work discussed in this paper addresses some of the above challenges. It elaborates on a Test Framework that includes test scenarios, test allocation, test result collection, analysis, and correlation of simulation and physical testing results, and evidence gathering. The structure of this paper is given in a spiral fashion. It starts at high level with all the elements in a scenario-based V&V process, and subsequently the workflows for both ADS V&V and simulation against real world comparison are illustrated. The paper then focusses on the simulation-based V&V process and demonstrates a detailed modularized framework. In the final part, the paper discusses how such framework is developed and implemented within the OmniCAV project. While the framework discussed in this paper is demonstrated from an ADS' perspective, the framework is relevant for other domains also.

II. Elements of the Evaluation Continuum

It is important to first differentiate and better understand the terms used in this paper, i.e., use case, test scenario and test case. A use case describes the system behavior as a sequence of actions linking the result to a particular actor. A test scenario is a specific path through a use case, i.e., a specific sequence of actions. A test case is a set of test case preconditions, inputs, and expected results, developed to drive the execution of a test item to meet test objectives, including correct implementation, error identification, checking quality, and other valued information [12]. A use case can correlate to multiple test scenarios, and a test scenario can result into multiple test cases. Similar concept is also proposed in a later study in which three levels of scenarios were proposed: functional scenario, abstract scenarios, logical scenario, and concrete scenario. Functional scenario sits at the most abstract level and can result into multiple logical scenarios, and one logical scenario can result into multiple concrete scenarios. Logical scenario describes parameter using ranges, and concrete scenario uses concrete values [13][14].

Figure 1 illustrates the key elements of a scenario-based evaluation continuum. The workflow is independent from the test execution environment, and is be applicable for simulation run, real-world execution as well as X-in-the-loop (XiL) testing. The core aspect of this workflow is the **scenario**; information required within a scenario is created, processed and assessed throughout the whole process. This forms the overall scenario-based V&V framework. At the high level, every scenario-based V&V framework will consist of three main elements: **scenario**, the (test) **environment** and **certification/safety evidence & argument** that are described below.

Scenario element sits at the upstream of the workflow, and from it the structured scenario artefacts together with pass/fail criteria are created. Scenario element includes three sub-processes: *"create"*, *"format"* and *"store"*. Create sub-process represents the creation of scenario content; scenarios can be created using two different approaches – knowledge-driven and data-driven. The OmniCAV project has explored various scenario generation methods belong to both approaches, this will be described in the following section in more details. Furthermore, the scenario creation can be tailored towards different focuses areas, for example system engineering, safety,

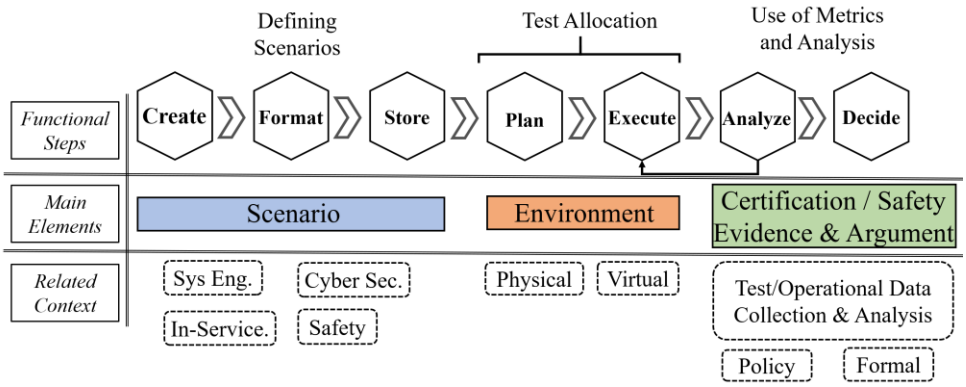

Figure 1: Elements of scenario-based workflow

cyber security or in-service testing. After scenario generation, the scenario content needs to be represented in a human and machine-readable format. A two-abstraction approach for scenario description [6] was used in the OmniCAV project which meet both regulatory and development needs of ADS testing. The structured scenarios will then be stored in a scenario database for storage, sharing, analysis and query purposes.

Environment element contains different choices of execution environment, such as real-world, simulation or a hybrid (XiL). Test allocation is a key step within the Environment element. This steps entails the allocation of test scenarios to be executed in different environments. Once the allocation or the test plan has been created, the next step is to execute and run the scenario.

Certification/safety evidence & argument element contains Analyze and Decide. Analyze can be further divided into various stages: 1) execution - whether the intended test case has been executed? 2) pass/fail assessment – monitoring the execution of the scenario and assessing the runtime output against a set of pre-defined pass/fail criteria/metrics, 3) scenario parameter space exploration – based on the current and past concrete parameters (e.g., speed, acceleration) and the pass/fail criteria, and a test case generator such as optimization algorithms that can be applied to introduce a new set of test case parameters with the aim of violating the scenario pass criteria. The output from the test case generator will result in the creation of new test cases and can then be fed back into execution module. This allows the increase of scenario coverage, the decrease of the 'unknow unsafe' region and the addition of new test cases into the database. The final stage is the Decide stage, based on whether the intended test cases have occurred, the assessment on the pass/fail criteria and the scenario coverage. were achieved. This stage will determine the output of the whole V&V process.

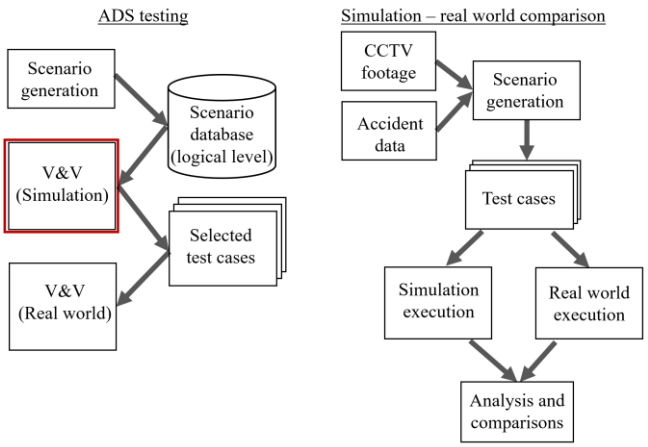

Figure 2: ADS testing flow, and simulation against real world comparison workflow

III. Validating Test Environment and Testing ADS

Although the main focus of this paper is to illustrate the testing workflow of the ADSs, the comparison of simulation against real world has also been explored. The purpose of the ADS testing is to test the ADS irrespective to its test environment. Given that simulation will play a key role in ADS testing, it is essential to also validate simulation as a test environment, in order to have confidence from the results of the ADS testing using simulation. The purpose of the simulation test against real world test comparison is to investigate the representativeness of the simulation environment. Therefore, the V&V process needs to consider both ADS testing and validation of the simulation (i.e. simulation and real-world comparison). Both these aspects can further be divided into activities based in physical environment and activities based in virtual environment.

Figure 2 illustrates the high-level workflow for both ADS testing and validation of simulation (i.e. simulation against real world comparison). Within the ADS testing, several scenario generation methods may be used for generating test scenarios that can result in a large number of scenarios for the tests. To effectively carry out large scale testing, first the simulation-based testing must be conducted, then a smaller set of selected scenarios for real world testing may be selected. One of the assumptions made within the ADS testing workflow is that '*simulation and real-world environments are comparable*', and hence simulation can be utilized to act as a filter and explorer for the scenario parameter space in order to provide the inputs to more economically expensive and risk-bearing real-world testing.

In the OmniCAV project, for simulation against real world comparison, a number of scenarios are generated using CCTV footage and accident data analysis. These scenarios represent the most common hazardous situations a vehicle could

encounter. This set of scenarios are then concretized to generate test cases and executed in both simulation and real-world environments. It should be noted that for ADS testing different execution environments are used in a sequential order, whereas for the validation of the simulation, they are executed in parallel. To replicate the real world environment, environmental data were collected within a combined rural/urban road loop in Oxfordshire in the UK [15], and a digital twin was created for simulation execution. During the execution, the same format of scenario data was received from both environments and were then compared. For rest of this paper, the simulation-based V&V workflow depicted in Figure 2 will be illustrated in further detail. This functional block will be expanded into a complete workflow at both functional and implementation levels.

IV. Simulation-based V&V Workflow

Figure 3 illustrates the logic flow of the simulation-based V&V process in terms of functionalities that have been developed in the OmniCAV project. Scenarios are generated and described using a human and machine-readable format at the logical scenario level. They are stored in the Safety Pool™ scenario database [16] and ready for query for testing via API. A scenario selector is implemented for performing such API calls. It iterates within a specific scenario library and retrieves individual logical scenarios. The test case generator is then used to generate test case parameters and optionally convert the test case into other desired executable formats. Upon execution, the test case data is processed and checked against three decision modules. The first one is whether the intended test case situation occurred. If *yes*, then the test case pass/fail criteria is checked. If *no*, then the test case run is checked against a pre-defined maximum iteration number of test cases. The test case pass/fail criteria module consists multiple types of criteria sources. If a test case fails the criteria then its parameters combination is recorded and the current logical scenario testing is terminated. If a test case passes, then it will be checked against the maximum iteration limit. In the last step, if the maximum iteration is not reached, the current test case parameters together with the pass criteria will be fed into test case generator where algorithms such as Bayesian optimization [9] (used in OmniCAV project) can be applied as *"concretizer"* to introduce new parameters with the goal of violating the pass criteria. The closed-loop formed by test case generator, the test execution and the three test case checks enable the exploration the parameter space set out within the logical scenarios, while increasing the test coverage and reduce the 'unknown unsafe' case.

A. Scenario generation

Currently, there are two different approaches for scenario generation: knowledge-driven and data-driven [5]. A knowledge-driven scenario generation approach utilizes domain specific knowledge to identify hazardous events systematically and create scenarios. A data driven approach utilizes the available data to identify and classify occurring scenarios. Eight different scenario generation approaches have additionally been investigated, as shown in Figure 4. Three of them are currently implemented in the OmniCAV project (options 1, 2, and 3 illustrated in Figure 3), and the other five

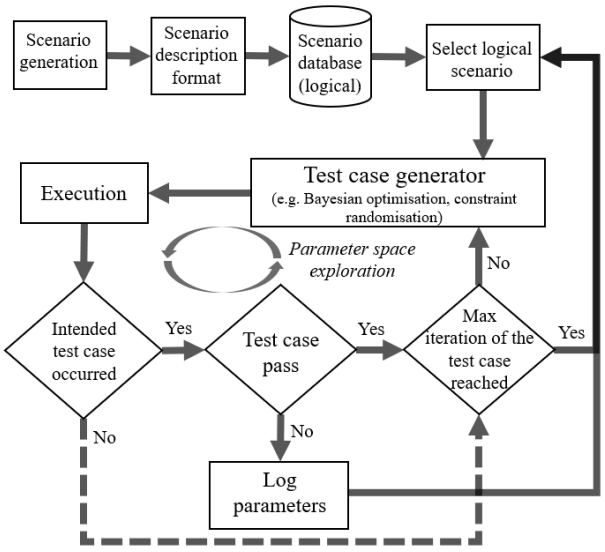

Figure 3: Logical flow of the simulation-based V&V process

methods have further been developed as part of a safety assurance framework for ADS. Since the focus of this paper is on the framework and its architectural implementation, all the individual elements such as scenario generation have been introduced at a high level, For further details can be found in the references provided.

For option 1, the publicly available STAT19 [17] UK accident dataset was analyzed to identify accident hotspots and scenario parameters which contribute to causation of accidents with carrying high levels of severity [15]. For option 2, anonymized insurance claim records provided by one of the OmniCAV project partners (Admiral) were also analyzed to identify the trends in near-miss events that lead to insurance claims [18]. For option 3, an extension to the Systems Theoretic Process Analysis (STPA) method was used to analyze the characteristics of the ADS architecture and identify system failures and hazardous situations [19]. The analysis was then converted into a set of logical scenarios together with their corresponding pass/fail criteria.

In addition to the options 1, 2 and 3, options 4 to 8 were have been explored and included in the framework. Option 4 uses the formal analysis approach with the highway code rules for scenario generation. Each of the highway code rules describes a hypothetical driving scenario with the corresponding behavior and ODD (Operational Design Domain) elements. The ODD is a specification set out by the manufacture of an ADS and it defines the operating conditions within which the ADS can operate safely [20]. Formal models are generated via a model template to create the mathematical representations of those scenarios, collecting the combinations of ODD and behavior parameters. The analysis reports the maneuver parameters that are near the boundary of violation, and produce scenarios that represent these set of violations. Option 5 uses similar formal representation as of option 4, but applied to the ODD of the ADS.

To effectively test the system against its defined ODD, only the boundary cases will be selected as compared to the whole ODD. In order to achieve this, the ODD specification is

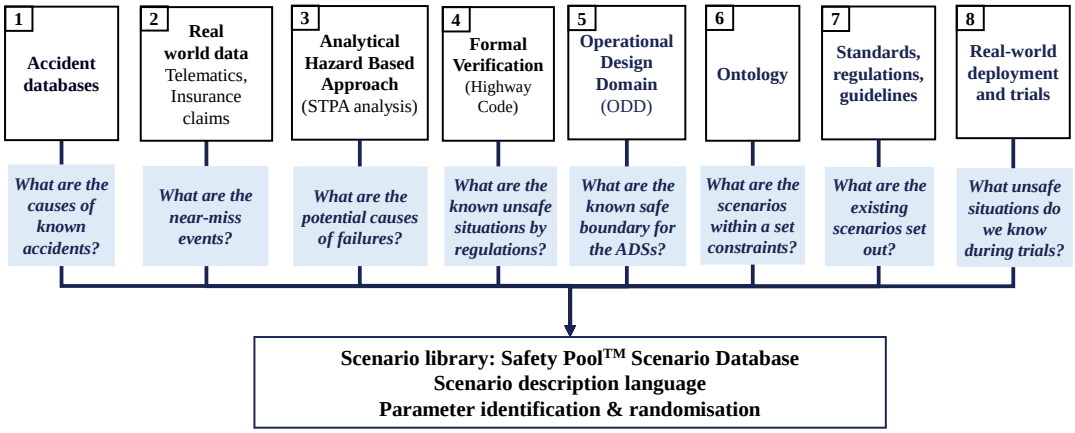

Figure 4: Various scenario generation methods explored

parameterized and represented by a formal model. Then the parameter combinations that form the ODD boundary that trigger a Minimal Risk Manoeuvre (MRM) or a transition demand can be extracted and result in a set of scenarios.

Inspired by the works in [4][5], option 6 uses an ontology-based scenario generation approach. Within this paper, ontology-based approach is proposed to be used to generate multiple similar scenarios from one set of input. An ontology defines all the classes within the domain. It also includes all the relationships between classes and all the pre-defined rules. An example rule can be *'if road A is connected to road B, then the width of road A must equal to the width of road B'*, such rules will ensure the correct instantiation of a scenario. By using: 1) a well-developed ontology (with the associated rules and properties); and 2) highly abstract scenario description of interest at the function level or a set of conditions, the abstract information can be detailed and used to generate large number of logical scenarios that can satisfy the initial conditions.

In addition to the above scenario generation methods, the existing scenarios already defined in the standards, regulations or guidelines (option 7) can also be utilized for the testing of ADSs, for example the scenarios set out in ISO22737 [21] and EuroNCAP [22]. ISO22737 has been developed for low-speed automated driving systems (LSAD) and the EuroNCAP provides a set of testing scenarios for the safety assurance of vehicles. An example of EuroNCAP scenario converted into logical scenario was previously illustrated in this paper section IV-B [6]. Option 8 includes the scenarios that occur during real world trials and deployments. Such scenarios might have not been considered pre-deployment, but are key learnings.

B. Scenario description language and sceanrio database

After generating the scenario content, an adequate scenario description language (SDL) is used to represent the content and enable its sharing and execution. A two-level abstraction approach of SDL, as depicted in Figure 5, has been previously developed and published as part of the OmniCAV project [6]. It was developed after analyzing the inputs received from various stakeholders such as AV developers, test engineers, regulators. SDL level 1 sits at the functional scenario level and is more abstract and its syntax resembles a structured natural language format. SDL level 2 sits at the logical and concrete scenario levels. It uses a formal machine-readable format, as shown in Figure 5. By additional detailing, one can convert SDL level 1 into level 2, and by abstracting the opposite can be achieved. In addition SDL level 2 can be converted to other ASAM OpenX formats for wider tool support.

The basic content of the SDL covers the scenery aspect, the environmental conditions, and the behavior aspect of the non-*Ego* agents; here the *Ego* refers to the vehicle under test. The concepts that cover the scenery and environmental conditions are referenced to the *BSI PAS 1883 (Operational Design Domain (ODD) taxonomy)* [20]. For the behavior aspect of the SDL, it is divided into maneuvers and agent type. Maneuvers include relative maneuvers and absolute maneuvers. Relative maneuvers indicate relations between multiple dynamic actors such as pedestrians, vehicles; such maneuvers include cutting in, moving towards which required two actors. Absolute maneuvers can be applied to a single actor, such as drive, turn right, etc. Agent type includes road users, pedestrians, and animals. For the scenery aspects, SDL considers any scenery settings as a roads-and-junctions network. Each road or junction is described individually using the types and the associated ODD attributes. In addition, for each junction, the connecting roads and lanes as well as connecting angles are also required, this can be referenced to the individual road description and allow the composition of the entire scenery. For the behavior description, the overall structure contains two parts: initialization phase and maneuvers phases. Initialization phase sets out the initial road and lane for each actor, the relative heading angles and relative positions between actors can also be defined. For the maneuver phases, a behavior tree

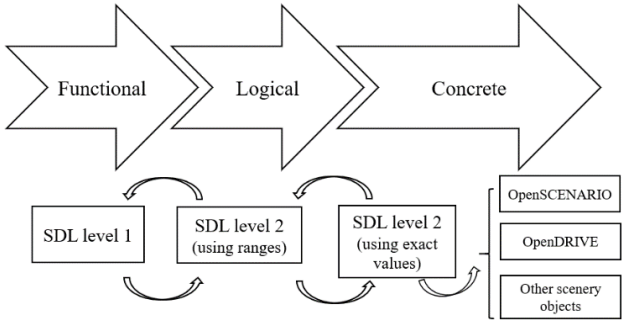

Figure 5: Two-level scenario description language mapped to the scenario abstraction levels

style description format is utilized. Each actor contains multiple activity phases in a sequential relation; when two actors are performing activities at the same time the activity phases between the two actors are in a parallel relation. Each activity phase consists of the actual maneuver activity and a trigger condition. Figure 6 illustrates the logic of an SDL behavior description consists of three actors, the first two actors each has two activity phases and these two actors are performing actions at the same time. The third actor starts to perform activities after the first two actors have stopped for the remaining of the scenario. The environment part of the SDL is treated as global ambient conditions that apply throughout the scenario. The list of all the values for the environment related ODD attributes within the SDL are also defined.

After creating the scenarios with SDL format, the next stage is to store them in a scenario database, which can be used by individuals and organizations to exchange, host, query and analyze them. In OmniCAV project, all the generated scenarios (~100,000) have been hosted in the Safety Pool™ Database. Scenario labels are used for query and analysis providing roles of tagging the real-world route with individual ODD labels and API connection and scenario visualization.

C. Test case generator and execution

Once the scenarios have been populated into the database, a scenario selector retrieves the relevant scenarios iteratively via API calls from the database, each individual logical scenario is then passed into the test case generator and starts the testing cycle. The test case generator can be divided into two different settings: the initial iteration, and subsequent iterations. During the initial iteration, the concrete test case parameters are instantiated using the average values of each value range defined in the OmniCAV project case. For the subsequent iterations, optimization algorithms can be implemented to intelligently select the parameter combinations for the next test case based on the test case assessment. A Bayesian optimization algorithm is implemented in the OmniCAV project, it is developed further based on the study given in [9], in which it investigated using Bayesian optimization and STPA scenarios to explore the unknown unknowns. From the test case analysis, both the scenario variables and the pass/fail assessment are provided to the test case generator where the optimization algorithm is embedded. The newly identified values create the next test case with the optimization goal of driving the system to violate its safe boundaries. It should be noted that the Bayesian optimization is only one example of the test case generator, one could fit other algorithms to generate new test cases, such as constraint randomization. Upon generating the test case, the next step is to run it in the simulation environment. In the OmniCAV project, Unity-based simulator developed by Thales UK-XPI is used for the environment simulation. The real-world generated map containing parts of Oxfordshire was developed for the XPI simulator. The simulator was then integrated with the ADS, the traffic simulator, and the test case analysis engine.

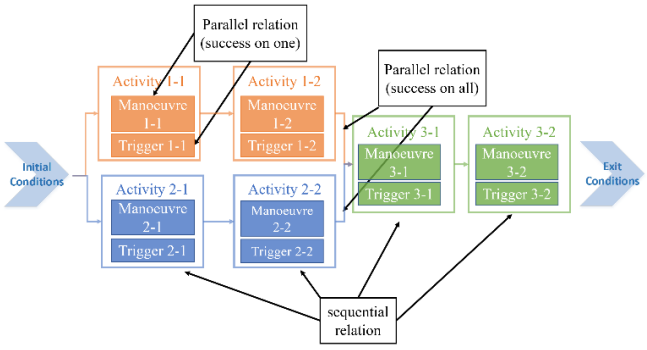

Figure 6: Example logics of an SDL behavior description

D. Test case analysis

1) Is the intended test case occurred?

The first step of the test case analysis is assessing whether the intended test case has taken place. The SDL behavior element is constructed and implemented using a behavior tree style. This provides the means for monitoring and assessment of the test case progress. Figure 7 illustrates an example behavior tree that consists of both parallel and sequential activities.

On the main branch, it has *Initialization*, *Manoeuvre1* and *Maneuver_set* in a sequence. Within *Manoeuvre1*, it has *action1* and *exit_cond1* in parallel relation with a success on one criteria. This means whichever node within the parallel relation finishes this tree branch, will succeed. Within *Maneuver_set* branch, it contains the maneuvers for both actor 1 and actor 2, along with their exit conditions. By using such behavior tree implementation, it provides the ability to assess which node has been completed, terminated, failed or in-progress. In the example, *Initialization* is completed and *Maneuver1* is completed as well by the *exit_cond1* being satisfied. However, within *Maneuver_set* the *actor2_manoeuvre* is failed due to its exit condition failed. Such information generated by the behavior tree implementation is used to assess whether the intended scenario has occurred.

2) Is test case passed/failed?

If the previous assessment result is *yes*, the test case data will then be passed to the *"pass/fail assessment function"*. However, if the assessment result is *no*, the test case will be checked against a pre-defined iteration limit. For the pass/fail assessment, several different types of criteria can be used including: STPA related, Highway Code derived, ODD related

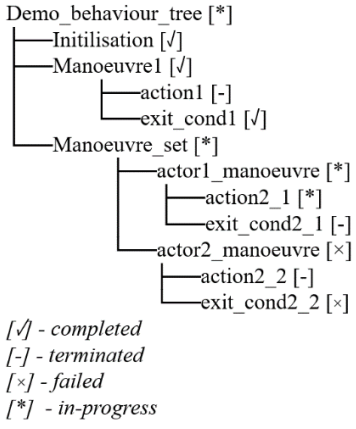

Figure 7: Example behavior tree status of a test case

or a set of generic criteria. Each of the STPA scenarios has its own specifically tailored pass criteria, however such criteria need to be converted into a machine-readable format and made accessible to the ADS. In [19], authors illustrate detailed insight into STPA-based scenario generation and evaluation.

The ODD based evaluation method uses the ODD specification of the ADS to form a safe operating boundary and evaluates the simulation ground truth data against the boundary during runtime. At any given point, the ADS could be inside or outside of its ODD, this can then be used to assess its ability to maintain within its ODD. In order to represent such boundary, a human readable and machine readable language was developed [23], the domain attributes used within the language were referenced to the BSI PAS 1883 ODD taxonomy [20]. The ground truth data is retrieved from the simulator during runtime, such ground truth is then filtered to only contain the attributes listed in the ODD taxonomy, and subsequently converted into a common intermediate format. On the other hand, the ODD specification is parsed and converted into the same intermediate format. An ODD assessment module is then implemented to compare the two intermediate formats derived from the ground truth data and ODD specification. In the OmniCAV project, an ontology-based assessment method is implemented, it utilizes open-source ontology reasoner to infer inside/outside of ODD. The UK highway code converted Digital Highway Code (DHC) is developed to serve as an oracle and evaluate the ADS' ability to obey the regulations. The DHC model contains ODD elements as well as the behavior elements, with each individual Highway Code rule being analyzed and converted into a quantifiable format. Ontology framework is used to represent the domain model and the rules, and ontology reasoner is used to perform runtime assessment of the ADS. In addition to the STPA-based, ODD-based and DHC-based test case pass/fail assessment, a set of generic assessment criteria are also used in the OminCAV project. For example, a fixed timeout is used to terminate test cases if needed. Collision criteria are also implemented to fail test cases whenever a collision of the vehicle is detected. Other criteria such as lane keeping, average speed limit, and maximum speed limit are also included.

3) Is maximum iteration of the test case reached?

Based on the pass/fail assessment, if the result is *yes*, the workflow is then sent to check whether the maximum iteration for the test case has reached. If the result is *no*, then the current test case parameter combination will be logged and the testing of the current scenario will be terminated. The maximum iteration is set to stop large number of loops for the test case generation within the same logical scenario. In our case, a hard limit is implemented across all the logical scenarios.

V. Implementation Architecture of the V&V Workflow

So far the functionality of the V&V workflow has been introduced. This section illustrates the implementation of the workflow. As shown in Figure 8, the necessary functions are modularized into four parts: Test manager, Simulation, Test Case Generator, and Test Case Analyzer. The Test Manager is in charge of orchestrating the whole workflow, all the communications are established between other modules and Test Manager. Simulation contains the environment simulator, the traffic simulator and the ADS under test. The Test Case Generator is for generating concrete test case parameters and optionally converts into other scenario format prior to execution. The Test Case Analyzer contains test case indexing – in order to obtain the scenario parameter variables and test case evaluation – for checking 1) whether intended test case occurred, 2) pass/fail assessment, and 3) whether maximum test case iteration is reached. The whole workflow is initiated by the Test Manager which is integrated with the scenario database for selecting the logical scenario and pass the scenario information to the Test case generator via scripts. Concrete test case is then generated by using the average values of the parameter value ranges, and optionally converted into other desired scenario formats before being sent back to the Test Manager. Meanwhile Test Manager also sends the scenario information to the Test Case Analyzer for indexing via protobuf (Protocol Buffers as a method of serializing structured data). Upon receiving the generated test case, Test manager sends the converted test case to Simulation using protobuf, it then sends a run signal to start the simulation. During runtime, live data is communicated between the Simulation and Test case evaluation module via protobuf. The Test case evaluation will then produce the evaluation results and send back to Test Manager. Test Manager then stops the simulation run, and checks the evaluation results against the three decision boxes. Based on the outcome, the Test Manager: 1) select a new logical scenario to test, 2) command to generate new test cases under the same logical scenario. Finally, Figure 9 shows a tool

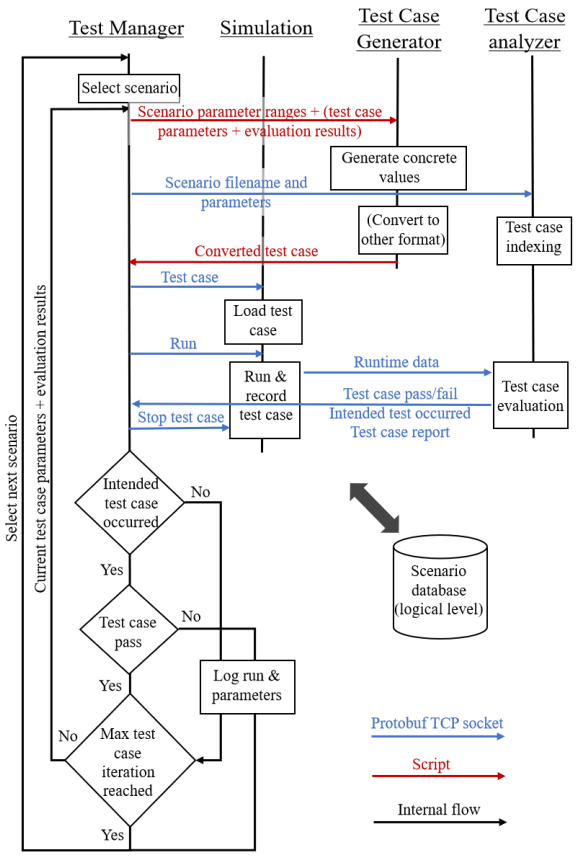

Figure 8: Implementation architecture of the workflow

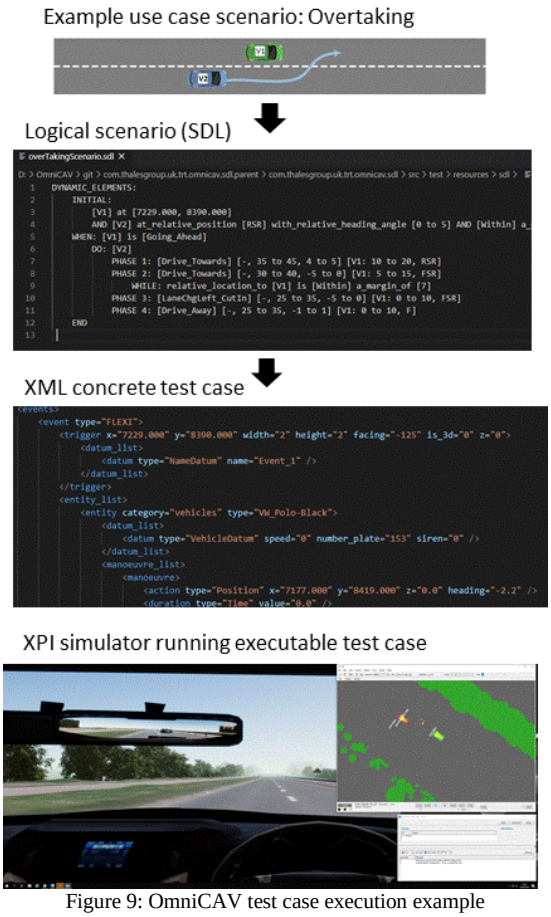

Figure 9: OmniCAV test case execution example

chain developed executing scenarios and test cases in the OmniCAV project.

VI. Conclusion

In this paper, a number of challenges are presented for V&V of Autonomous Systems. To address some of the main challenges, we introduced a scalable and domain agnostic V&V framework, demonstrated in an ADS context. This framework covered a number of key elements for conducting V&V tests in both simulation and physical environments. The paper mainly illustrated the details of the simulation-based V&V process and covered scenario generation, scenario description format, test case generation, test case evaluation methods and parameter space exploration methods, all with the aim of gathering evidence for both functional and behavioral verification. Using the framework, we developed and implemented the framework into a toolchain for practically conducting the V&V tests.

Acknowledgment

The work presented in this paper has been carried under the Innovate UK and Centre for Connected and Autonomous Vehicles (CCAV) funded OmniCAV project (Grant No. 104529). This work is also supported by UKRI Future Leaders Fellowship (Grant MR/S035176/1). The authors would like to thank their colleagues at Thales UK RTI, XPI, WMG, Aimsun and other OmniCAV project partners for their contributions in the development of this work.

References

[1] D. Hond, A. White, and H. Asgari, “Quantifying Dataset for Systematic Artificial Neural Network Classifier Verification,” *Proceeding Safety-Critical Syst. Symp.*, 2011.

[2] N. Kalra and S. M. Paddock, “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?,” *Transp. Res. Part A Policy Pr.*, vol. 94, pp. 182–193, 2016.

[3] S. Khastgir, S. Brewerton, J. Thomas, and P. Jennings, “Systems Approach to Creating Test Scenarios for Automated Driving Systems,” *Reliab. Eng. Syst. Saf.*, 2021.

[4] G. Bagschik, T. Menzel, and M. Maurer, “Ontology based Scene Creation for the Development of Automated Vehicles,” *2018 IEEE Intell. Veh. Symp.*, no. Iv, pp. 1813–1820, 2018.

[5] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, “From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment,” *IEEE Intell. Veh. Symp. Proc.*, vol. 2019-June, pp. 2383–2390, 2019.

[6] X. Zhang, S. Khastgir, and P. Jennings, “Scenario Description Language for Automated Driving Systems: A Two Level Abstraction Approach,” in *Proc. of the 2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2020.

[7] D. Fremont *et al.*, “Scenic: Language-Based Scene Generation,” *UC Berkeley EECS Tech. Rep.*, 2018.

[8] ASAM e.V., “ASAM OpenSCENARIO,” 2020. [Online]. Available: https://www.asam.net/standards/detail/openscenario/. [Accessed: 30-Apr-2020].

[9] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, “Identification of Test Cases for Automated Driving Systems Using Bayesian Optimization,” pp. 1961–1967, 2019.

[10] S. Ulbrich, T. Nothdurft, M. Maurer, and P. Hecker, “Graph-based context representation, environment modeling and information aggregation for automated driving,” *IEEE Intell. Veh. Symp. Proc.*, no. Iv, pp. 541–547, 2014.

[11] E. de Gelder *et al.*, “Ontology for Scenarios for the Assessment of Automated Vehicles,” 2020.

[12] S. Khastgir, G. Dhadyalla, S. Birrell, S. Redmond, R. Addinall, and P. Jennings, “Test Scenario Generation for Driving Simulators Using Constrained Randomization Technique,” in *SAE Technical Paper# 2017-01-1672*, 2017.

[13] T. Menzel, G. Bagschik, and A. M. Maurer, “Scenarios for Development, Test and Validation of Automated Vehicles,” *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, no. Iv, pp. 1821–1827, 2018.

[14] 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.

[15] M. Brackstone *et al.*, “OmniCAV: A Simulation and Modelling System that enables CAVs for AI,” *IEEE Intell. Transp. Syst. Conf.*, vol. 2020, pp. 2190–2195, 2019.

[16] “Safety Pool Scenario Database.” .

[17] “Road Safety Data - STATS19,” *UK Department for Transport*, 2020. [Online]. Available: https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data. [Accessed: 02-Apr-2020].

[18] E. Esenturk, S. Khastgir, A. Wallace, and P. Jennings, “Analyzing Real-world Accidents for Test Scenario Generation for Automated Vehicles *,” in *IEEE Intelligent Vehicles Symposium 2021 (submitted for review)*.

[19] S. Chen, S. Khastgir, I. Babaev, and P. Jennings, “Identifying Accident Causes of Driver-Vehicle Interactions Using System Theoretic Process Analysis (STPA),” in *Proc. of the 2020 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2020, pp. 3247–3253.

[20] “Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification,” *The British Standards Institution, BSI PAS 1883*. 2020.

[21] ISO, “Intelligent transport systems — Low-Speed Automated Driving (LSAD) Systems for Predefined routes — Performance requirements, system requirements and performance test procedures - ISO 22737,” 2021. [Online]. Available: https://www.iso.org/standard/73767.html.

[22] Euro NCAP, “Euro NCAP 2025 Roadmap: In pursuit of Vision Zero,” 2017.

[23] P. Irvine, X. Zhang, S. Khastgir, and P. Jennings, “A Textual Description Language for the Operational Design Domain of Automated Driving Systems,” *IEEE ITSC*, 2021.