Scenario Description Language for Automated Driving Systems: A Two Level Abstraction Approach

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

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

***Abstract*—The complexities associated with Automated Driving Systems (ADSs) and their interaction with the environment pose a challenge for their safety evaluation. Number of miles driven has been suggested as one of the metrics to demonstrate technological maturity. However, the experiences or the scenarios encountered by the ADSs is a more meaningful metric, and has led to a shift to scenario-based testing approach in the automotive industry and research community. Variety of scenario generation techniques have been advocated, including real-world data analysis, accident data analysis and via systems hazard analysis. While scenario generation can be done via these methods, there is a need for a scenario description language format which enables the exchange of scenarios between diverse stakeholders (as part of the systems engineering lifecycle) with varied usage requirements. In this paper, we propose a two-level abstraction approach to scenario description language (SDL) – SDL level 1 and SDL level 2. SDL level 1 is a textual description of the scenario at a higher abstraction level to be used by regulators or system engineers. SDL level 2 is a formal machine-readable language which is ingested by testing platform e.g. simulation or test track. One can transform a scenario in SDL level 1 into SDL level 2 by adding more details or from SDL level 2 to SDL level 1 by abstracting.**

Keywords— scenario, testing, scenario definition language, verification and validation, automated driving systems.

I. INTRODUCTION

Recent years have seen an increasing deployment of Cyber-physical systems (CPSs) in the society. CPSs are an integration of computation and physical systems where embedded controllers control physical processes [1]. Building safe CPSs offers variety of challenges with a key challenge being the need for systems approach to the design. Moreover, there is a major temporal attribute to the functionality for safety of safety critical CPSs. Automated Driving Systems (ADSs) and Advanced Driver Assistance Systems (ADASs) are two types of CPS which have received an increasing focus in the automotive industry. The move to CPSs is driven by their many potential benefits, ADASs and ADSs in particular offer benefits such as improving safety [2], lowering vehicular emissions [3], improving traffic throughput [4], and decreasing drivers' workload [5]. However, the increased complexity of these CPSs due to the system interactions and their temporal nature, have manifested into testing challenges for safe introduction of the technology [6], [7].

ADASs and ADSs are typically developed based on a process derived from the V-model, which represents a system development lifecycle that consists several development steps and their corresponding testing counterparts [8]. The development steps form one side of the 'V' shape, which starts from high level requirements to systems development and further to low level module development. The testing phases form the other side of the 'V', which ranges from module level unit testing to system testing and to user acceptance testing. The V-model allows validation and verification activities to be carried out across the system development cycle to ensure the quality and correctness of a system. For ADSs, Kalra et. al suggested that it would need to be driven for 11 billion miles to demonstrate they are 20% better than human drivers [9]. While this might seem to be an unfeasible proposition, a more meaningful metric should be the types of scenarios experienced by the systems, leading to scenario based testing [10]. EuroNCAP, the European car safety assessment programme in their 2025 roadmap has also highlighted the role of scenario based testing in the safety evaluation of ADASs and ADSs [11]. Furthermore, for complex systems, the emergent behaviour due to sub-system interactions lead to occurrence of *"unknown unknown"* scenarios. Khastgir et. al. suggested for ADASs and ADSs, focus needs to be on *"how a system fails"* as compared to *"how a system works"* this leads to the focus on the scenarios which expose failures, i.e. Hazard Based Testing [12].

An important aspect of developing and storing test scenarios along the V model is the need for appreciation about the diversity of its end users (e.g. autonomous vehicle (AV) technology developer, simulation test engineer, real-world test engineer, regulators, public etc.). Each of these end users have varied requirements and at different levels of abstraction. An AV technology developer would favour a common structure for scenarios in order to reuse across systems and organizations, a test engineer would want a high level of specificity to have an objective understanding and be able to execute the scenario on a test platform (e.g. simulation platform, test track). A regulator would want a common structure at a higher abstraction level to enable non-specialists to understand the test scenarios. At the same time, a regulator would also want a common structure for test scenarios to enable them to compare scenarios from different manufacturers. While there are synergies in the requirements from the various end-users, some are also competing (e.g. executability vs high level of abstraction). Inspired by the functional scenario, logical scenario and concrete scenario concept, there is a need for a similar approach for different levels of abstraction for scenario definition language.

Manuscript 871 submitted to 2020 IEEE International Conference on Systems, Man and Cybernetics (SMC). Received May 29, 2020.

The creation of scenarios is usually based on expert knowledge or various data sources, and the execution of scenarios can be realised in X-in-the-loop (XiL) simulations or real-world testing. However, between the creation and execution of scenarios, there lacks a common, understandable and executable language to describe and transfer the scenarios. Different simulators often use tool specific languages that are proprietary to their individual environment. Moreover the abstraction level(s) of the description languages have not yet reached agreement across industry and academia. This not only limits the sharing and exchange of scenarios, but often leads to misinterpretations. While defining the content of test scenarios has been a major focus of the industry up until now, relatively newer efforts have now begun in the area of the format for scenario definition (mostly for simulation use). These include OpenSCENARIO [13], Wise Drive [14], ontological approaches [15] etc.

II. Related Work

Across industry and academia, several definitions of scenario have been proposed previously. Go and Carroll [16] define scenario within the context of system design, as a description that contains actors, their environment and goals, and sequences of actions and events. Geyer et al. [17] then concluded that, under the context of ADSs, a scenario includes at least one situation within a scene and the ongoing activities of one or both actors. Later, Ulbrich et al. [18] introduced the definition of scenario for ADSs as follows:

'A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Action & events as well as goals & values may be specified to characterise this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time.'

A recent study from Gelder et al. [19] further defines a scenario as a description of the characteristics of the ego vehicle, its activities and/or goals, its environment, and all the events that are relevant to the ego vehicles. Within the definition, an activity describes the time evolution of state variables between two events, and an event marks the time instant at which system transits between modes. Rather than explicitly describing the scenes, as proposed by Ulbrich et al. [18], Gelder et al.'s proposal focuses on activities and events to implicitly describe the scenes. Based on the scenario definition by Ulbrich et al. [18], Menzel et al. [10] proposed an extension to include three different abstraction levels: functional scenarios, logical scenarios and concrete scenarios. Functional scenario represents the most abstract level, on a semantic format it includes a description of entities and relations/intersections of them. Logical scenario represents the functional scenario using state space variable ranges, and concrete scenario defines the exact values within the ranges to derive unique test case.

The creation of scenarios can be data-driven [20][21] and knowledge-driven [15]. A data-driven approach utilises the available data to identify occurring scenarios, the disadvantages of this approach are: 1) the data do not describe all aspects of a scenario, 2) requires knowledge input to define parameter dependencies, 3) limited reference to the operational design domain [22], 4) challenging to extract interesting test scenarios [20]. A knowledge-driven approach utilises expert knowledge to identify hazardous events systematically and create scenarios. However, a complete identification of all the scenarios cannot be guaranteed [15], and the generated cases might not be valid or representative in real-life [20].

To construct scenes within a scenario, a five-layer-model was used by Bagschik et al [15], this was further extended to a six-layer-model by Bock et al [23]. The first layer describes the road layout, the second layer contains the traffic infrastructure such as barriers and traffic signs. The third layer represents the temporary manipulations of the first two layers, such as road works or construction site. The fourth layer introduces the dynamic and stationary objects. The fifth layer provides environment information such as weather, lighting. And the sixth layer includes data and communications. Based on this layered model, the automation of scene creation was achieved by Bagschik et al [15]. Several forms of expression can be used to describe the elements within a scenario and their relations. Gronniger et al. [24] compared three possible forms of languages in the context of modelling: 1) textual languages, 2) graphical languages, and 3) a combination of the two. Comparing to the graphical format, textual languages offers better readability, higher efficiency, space-saving and easier integration[24][8]. In the field of ADSs, textual scenario descriptions or a combination of textual and graphical descriptions are used across different abstraction levels. When the two formats are used in combination, the textual description tends to be incomplete [8]. In addition, The quality, expressions, and semantics in a textual scenario description are highly dependent on the individual creator, ambiguities and misunderstandings often exist [25].

To tackle this problem, the industry and academia have been working towards a common, exchangeable and executable language to describe scenarios for ADSs. Several simulation platforms offer tool-specific languages which are unique to their software, such as [26][27], this raises challenges for sharing and integration of scenarios. Open language formats such as OpenStreetMap [28], OpenDRIVE [29] and Lanelets [30] are popular examples for describing the road layout and road features, which forms the first three layers of the six-layer-model. However, OpenStreetMap is mainly designed for geographical purposes and not for driving simulator or ADSs [31]. OpenDRIVE is more used in industry and lanelets is more popular in academia. One of the main benefits of OpenDRIVE is the exchangeability between simulators, on the other hand lanelets is more lightweight but still able to offer great level of details. Althoff et al. [31] developed a conversion tool between lanelets and OpenDRIVE format. Currently still under development, OpenSCENARIO [13] is an emerging format for the description of the dynamic and environment elements (fourth and fifth layers). The dynamic elements are described through actions and events. Menzel et al. [22] presented an automation process to detail a keyword-based scenario description for execution, both OpenDRIVE and OpenSCENARIO file formats were used to set the concrete scenario parameters and integrate to the simulator. Pilz et al. [32] used OpenSCENARIO format to construct the dynamic elements of a scenario, integrated and executed the OpenSCENARIO file in a simulator.

Manuscript 871 submitted to 2020 IEEE International Conference on Systems, Man and Cybernetics (SMC). Received May 29, 2020.

Although the methodologies for scenarios creation exists, a commonly recognised language that can cover all aspects of a scenario is still missing, this limits the reproducibility of scenarios and the exchangeability between tools [33]. Moreover, the language shall cover different abstraction levels, from functional level to concrete level, to address different audience groups.

TABLE I. SCENERY ELEMENT EXAMPLES FOR THE THREE SCENARIO LEVELS

Abstraction level	Example description
Functional scenarios	A three-lane motorway with a straight geometry
Logical scenarios	Road type [Motorway], number of lanes [3], lane width [3.5-4.5]m, road curvature [0-5] degree
Concrete scenarios	Road type [Motorway], number of lanes [3], lane width [4]m, road curvature [0] degree

III. DEVELOPMENT OF SCENARIO DESCRIPTION LANGUAGE

Fig. 1. SDL standard format example, which includes actors, actions, scenery and the overall context

To develop a description language that can be used throughout the development cycle of an ADS, the language needs to be divided into different levels based on the information details and data values. Inspired by the three abstraction levels of scenarios [10] as illustrated in Table I, two different levels have been proposed for the SDL, SDL level 1 at the functional level, and SDL level 2 at the logical and concrete level. Based on the elements described in the six-layer-model, SDL elements are split into the following three groups – dynamic elements, scenery elements and environment elements. Dynamic elements consist of actors and their corresponding actions, the actors include both ego vehicle and those who can influence ego vehicle directly or indirectly. Scenery elements consist of all the physically static elements in the scene, such as road layout, traffic lights etc. Environment elements define the physical conditions such as precipitation, lighting and connectivity. An SDL standard format can be formed by connecting the three groups together: **Actor** is doing **action** in **scenery** when context (Fig. 1).

A. Dynamic elements

The SDL dynamic elements provide description for the behaviours of all the moving objects in a scenario, which include road users, pedestrians and animals. There are two categories within the dynamic elements, scripted traffic and non-scripted traffic, which will be discussed in the following section.

1) Scripted traffic

For the scripted traffic, the full dynamic behaviour description is included in the scenario description. It is usually required for the non-ego actors which directly influence the behaviour of the ego vehicle. Two approaches have been used to define the dynamic behaviour of entities (vehicles, pedestrians etc.). The first approach uses relative manoeuvres in relation to other dynamic or scenery objects. And the second uses absolute manoeuvres that are independent from the rest of the scenario elements. Taking an overtaking and lane-change cut-in scenario as an example, under relative concept one would define the behaviour of one vehicle in relation to the other such that it could accommodate a range of different parameters and provide more flexibility. On the contrary, using the absolute concept one would need to pre-determine the parameters of both vehicles individually in a highly accurate manner such that the two vehicles can reach synchronisation. The absolute manoeuvre approach is susceptible to changes in the outside conditions and will require additional input to calculate the parameter values. Moreover, as the complexity level increases (such as increase in number of entities), scenarios based on absolute manoeuvres will become difficult to create, interpret and maintain, due to the difficulties on establishing the spatio-temporal relationships between entities. It is therefore important to combine relative manoeuvres with absolute manoeuvres when describing the dynamic behaviour of entities, this defines both the types of movement of the entity and its relations to other entities or objects. Each combined manoeuvre consists of initial conditions and exit conditions to set the start and end parameters, and at least one phase where it is executed. The number of phases depends on the complexity of the scenarios and the transition between phases can be initiated by triggers (e.g. time triggers, location triggers and condition triggers etc).

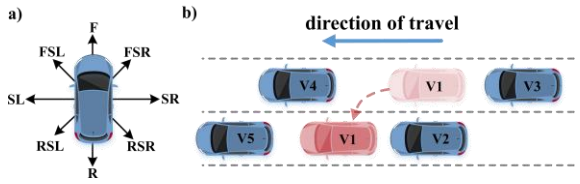

Fig. 2. a) vehicle relative position compass, b) example for relative manoeuvre illustration

In order to describe the dynamic behaviour of road users, a list of manoeuvre types is identified for both absolute and relative manoeuvres (Table II). Among the absolute manoeuvres, 'Drive' means the vehicle is moving forward, 'Stop' corresponds to a stationary vehicle, and lane change only takes places between current and adjacent lanes. For the relative manoeuvres, Fig. 2b. is used to illustrate the terms. It displays a 'lane change left' absolute manoeuvre performed by V1 within a two-lane road. In this example, V1's relative manoeuvre is 'cut-in' for both V5 and V2, and 'cut-out' for V4 and V3. Upon finishing the lane change manoeuvre, V1 is 'moving towards' V5 and V4, and 'moving away' from V2 and V3. By combining the absolute manoeuvres with the relative manoeuvres, thirty-six manoeuvre identities can be obtained to form the SDL road user manoeuvre matrix. For instance, *Drive + Moving Towards → Drive_Towards, Lane Change Right + Cut-in → LaneChangeRight_CutIn*.

TABLE II. ABSOLUTE AND RELATIVE MANUVER TYPES FOR ROAD USERS

Absolute maneuvers	Drive, Stop, Lane change right, Lane change left, Turn right, Turn left, Reverse, Miscellaneous, Collide
Relative maneuvers	Cut-in, Cut-out, Moving towards, Moving away

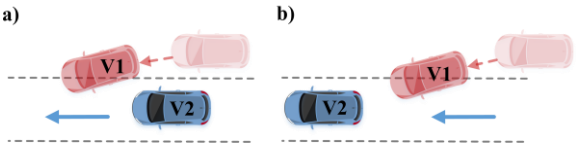

Fig. 3. Change lane left cut-in examples

However, the manoeuvre matrix on its own is not sufficient to describe the dynamic behaviour between two vehicles in a unique manner. For example, in Fig. 3 both a) and b) can be described as V1 changing lane left and cut into V2's lane. In order to differentiate between the two examples, V1's relative position to V2 must be taken into consideration. The vehicle's relative position compass is introduced as part of the manoeuvre description. It consists of eight different positions, as shown in Fig. 2a. Applying the compass to case a (Fig. 3) would mean that V1 is changing lane left cut-in with relative position FSR to V2, while with relative position RSR to V2 in case b.

After the main dynamic attributes (manoeuvre matrix, relative position, event triggers) are identified, the next step is to construct syntax for both initialisation and manoeuvre phases. Initialisation process will first require specifying the road and lane IDs for the agent vehicle (A_ID), the heading angle of the A_ID will be determined by the lane specification. To initialise a relative agent vehicle (RA_ID), its road and lane IDs need to be specified, in addition the relative position of RA_ID to A_ID also need to be specified. To construct the manoeuvre syntax, user first needs to specify a manoeuvre matrix ID (Func_Type) for the A_ID. Two direct inputs for the manoeuvres are speed and acceleration, this will be specified for the A_ID, in addition the relative speed of A_ID to RA_ID will also be specified. Please note the acceleration attribute is only effective when the target speed is not reached. Apart from the speed and acceleration, the syntax also contains position information such as relative position, junction information (whether vehicle is within a junction) and lateral / longitudinal margin. The position information serves as the triggers for entering and exiting manoeuvre phases. For instance, agent vehicle would perform a right turn only when the junction condition is satisfied. The syntax for both initialisation and manoeuvre is illustrated in Fig. 4.

a)
[A_ID] in [R_ID.L_ID] AND [RA_ID] in [R_ID.L_ID] at [R_P1] relative position
b)
[A_ID] [Obj] [Func_Type] [J, A_S, A] [RA_ID: R_S, R_P2]
WHILE: [A_ID] relative position to [RA_ID] is [Con] a [Dir] margin of [M]

A_ID: Agent Vehicle
R_ID: Road ID
L_ID: Lane ID
RA_ID: Relative agent vehicle
R_P1: RA_ID relative position to A_ID
R_A: RA_ID relative angle to A_ID
Obj *: Object that A_ID is colliding with
Func: Absolute maneuver types
Type: Relative maneuver types
J *: Junction where maneuver takes place
A_S: Absolute speed of A_ID
A: Absolute acceleration of A_ID
R_S: A_ID relative speed to RA_ID
R_P2: A_ID relative position to RA_ID
Con: Within/ not within
Dir: Lateral/ longitudinal
M: Margin
*Note: 1) * represents optional attributes, 2) the 'WHILE' condition in syntax b) is optional*

Fig. 4. SDL level 2 road users dynamic element syntax for a) initialisation, b) manoeuvre

[A_ID] [Obj] [Func_Type] [C, A_S] [RA_ID: M_A, R_P]

A_ID: Agent
Obj *: Any object that A_ID is colliding with
Func: Absolute maneuver types
Type: Relative maneuver types
C *: Crossing where maneuver takes place
A_S: Absolute speed of A_ID
RA_ID: Relative agent
M_A: A_ID relative maneuver angle to RA_ID
R_P: A_ID relative position to RA_ID
*Note: * represents optional attributes*

Fig. 5. SDL level 2 pedestrians/ animals dynamic element syntax

The executability of the syntax shown in Fig. 4 was evaluated using the open-source CARLA simulator [27]. An SDL python-based tool chain was developed to execute a scenario in SDL Level 2 format in the simulator. The tool chain consists of an SDL manoeuvre library and an auto-scripting / parsing library. Based on the scenarios defined in SDL level 2 format, the tool chain specifies parameter values within the defined range to create test cases values. The SDL manoeuvre library contains the execution code for all the manoeuvre identities in the SDL manoeuvre matrix. The auto-scripting/parsing library first generates the code for the temporal and spatial based event triggers, it then combines the triggers with the manoeuvre identities to complete the execution code for the dynamic element of a scenario. The feedback from the CARLA evaluation has been reflected in the syntax shown in Fig. 4.

Different from the SDL level 2 format, the SDL level 1 description is more textual and at higher abstraction level. Instead of using the manoeuvre IDs (absolute + relative manoeuvres), SDL level 1 separates the two terms to form a plain and spoken expression. Moreover, at level 1 the relative speed and acceleration values between vehicles will no longer be specified, it uses more general terms such as 'acceleration, constant, deceleration' to indicate (see case studies).

TABLE III. ABSOLUTE AND RELATIVE MANUVER TYPES FOR PEDESTRIANS AND ANIMALS

Absolute maneuvers	Stop, Walk forward, Walk backward, Turn right, Turn left, Turn backward, Run, Slide, Miscellaneous, Collide
Relative maneuvers	Moving towards, Moving away, Crossing agent's lane

Similar philosophy was used to develop the syntax for pedestrian / animal manoeuvre description. The absolute and relative manoeuvre types used in this case are illustrated in Table III. Combining the absolute and relative manoeuvre types results

in thirty manoeuvre identities for the pedestrian / animal manoeuvre matrix. For example, *walk forward + moving towards element → WalkForward_MovT*. The corresponding syntax is displayed in Fig. 5.

2) Non-scripted traffic

The increasing demand for more realistic simulations [13] to enhance the testing capabilities has raised the requirements of more intelligent actors to be represented in scenario executions and thus also in scenario definitions. To cater to this requirement, in a simulation setting, the non-scripted (goal based) traffic description is used to define intelligently controlled traffic, which can be based on AI decisions or surveillance cameras recordings replay. The attributes used to define non-scripted traffic include agent type, traffic density (agents/distance), traffic speed (distance/time), presence of special vehicles, and starting / finishing points.

B. Scenery elements

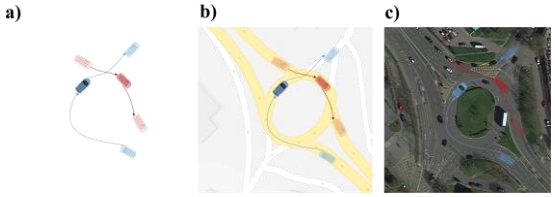

Fig. 6. Graphical representations of the scenery detail levels: a) no scenery elements, b) SDL level 1 representation, c) SDL level 2 representation

The scenery elements provide descriptions of the static elements within a scenario using junctions and roads as the building blocks. Fig. 6 illustrates the graphical representations for the two different levels of the scenery elements. Figure 6a displays two vehicles manoeuvre within an empty space without any defined scenery elements. Figure 6b provides road geometries and junction connections, this corresponds to the concept of SDL level 1. Figure 6c adds more elements to the scenery such as lane specifications, road structures, traffic control, roundabout details etc, this corresponds to the detail level of SDL level 2. In theory, SDL level 2 scenery elements could contain large number of attributes in order to represent real world scenarios. However, for a scenery to be relevant, it needs to be part of the Operational Design Domain (ODD) of the ADS. ODD refers to the operating conditions underwhich an ADS can perform safely [34]. In order to effectively describe the scenery while maintaining a compact format, the attribues from ODD taxonomy (ISO 34503) has been used [35], the high level elements are shown in Fig. 7. Zones specify any special road configurations which may differ from normal driving conditions, or area with special regulations. Drivable areas provide description of the elements that are directly related to the vehicle's manoeuvrability. The junction attribute consists of

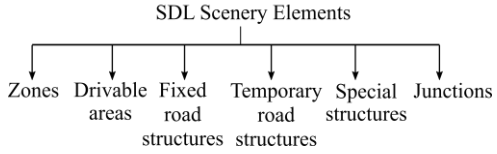

Fig. 7. High level categories of the SDL scenery elements

SDL level 1 description:
DO: Map-roads and junctions network [Network 1] as:
Junctions:
YJ1: Junction [Y-Junction] which has [No] connection control and [3] connections with [R1, R2, R3]
Roads:
R1: Road type [Motorway] with [Straight] geometry in [Urban] environment
R2: Road type [Motorway] with [Straight] geometry in [Urban] environment
R3: Road type [Slip road] with [Straight] geometry in [Urban] environment

SDL level 2 description:
DO: Map-roads and junctions network [Network 1] as:
Junctions:
YJ1: Junction type [Y-Junction] as [YJ1] which has [No] connection control and [3] connections with [R1, R2, R3]
- Angles between roads [R1 -> R3, R1 -> R2, R3 -> R2][145 to 155, 175 to 185, 25 to 35] degrees
- Road lane connections [R1.L1 -> R3.L1, R1.L1 -> R2.L1, R1.L2 -> R2.L2, R1.L3 -> R2.L3]
- Dimensions [Width: 10 to 12, Depth: 10 to 12]
- Road sign [N/A]
- Traffic light [N/A]

Roads:
R1: START
Road type [Motorway] as [R1] with zone as [N/A] AND speed limit of [70] in [Urban] environment
- Number of lanes [3] as [R1.L1, R1.L2, R1.L3]
- Road traffic direction [Left] with lane type [Traffic lane]
- Lane markings [Broken lines]
- Road surface type [Uniform] with surface condition [Dry] AND surface feature [N/A]
- Horizontal road geometry [Straight] with curvature radius of [N/A]
- Vertical road geometry [Level plane]
- Transverse road geometry [Undivided] with [No] road side features
- Length [200 to 500] AND lane width [4 to 4.2]

END: [YJ1]
R2: START: [YJ1]
Road type [Motorway] as [R2] with zone as [N/A] AND speed limit of [70] in [Urban] environment
- Number of lanes [3] as [R2.L1, R2.L2, R2.L3]
.....
- Length [200 to 500] AND lane width [4 to 4.2]

END
R3: START: [YJ1]
Road type [Slip road] as [R3] with zone as [N/A] AND speed limit of [70] in [Urban] environment
- Number of lanes [1] as [R3.L1]
....
- Length [100 to 200] AND lane width [4 to 4.2]

END

Fig. 8. SDL level 1 and level 2 scenery descriptions of a straight motorway with a Y junction

roundabouts and intersections. Road structures are classified into special road structures, fixed road structures and temporary road structures. The following example (Fig. 8) illustartes both SDL level 1 and level 2 format, it describes a straight motorway which connects to an exit slip road. Please note part of the descriptions for R2 and R3 are hidden due to repetitions and space constrain, a full description will expand the details.

C. Environment elements

The environment elements describe the physical conditions of the scenarios such as lighting, wind, cloudiness, etc. These characteristics are part of the ODD definition, hence the ODD Taxonomy [35] was used as a reference to determine the SDL attributes for the environment elements. These attributes can be divided into four main categories: weather, particulates, illumination and connectivity. Weather include wind, rainfall and snowfall, which are defined in m/s, mm/h and visibility (km)

respectively. Illumination include day light and artificial lighting, in the case of day light illumination the elevation angle of the Sun above the horizon and the relative position of the Sun need to be defined. An example of the environment element description can be found in case study 1.

IV. Case Studies

Two case studies are presented in this section to demonstrate the key properties -- common structure, executability and understandability – of SDL. The first case study utilises the STATS19 accident dataset, converts the relevant parameters into scenarios using SDL format, which can be further used for execution or sharing. The second case study creates SDL scenario descriptions based on the Euro NCAP test scenarios.

A. Case study 1: Formating accident data into SDL

For the first case study, the publicly available accident dataset STATS19 were used [36]. The original dataset contains sixty-seven variables, among which thirty-two variables describe the accident circumstances, which equivalent to the SDL scenery and environment elements. The next twenty-one of the variables describe the details of the vehicle and driver, together with high level vehicle manoeuvre description, this can be related to the SDL dynamic element. The last seventeen of the variables describe the casualty, this is not part of the SDL elements. After filtering through the parameters, fourteen most relevant variables were used as the input for generating the SDL format. Table IV displays the filtered STATS19 parameters for a selected example, it can be seen that most of the parameters contribute to the scenery elements, the remaining contribute to the environment and dynamic elements. Among all the variables, 1st road class corresponds to the road where the accident occurred, junction details indicate the nearest junction, and 2nd road class indicates the nearest road which the junction connects to. Translating these accident parameters into spoken language will result in the following description:

On a clear sunny morning with no high winds, on-road vehicle is performing an overtaking in an adjacent lane to ego vehicle at a dual carriageway motorway. The road surface condition is dry and the speed limit is 70 mph, there is no junction within 20m of range.

TABLE IV. STATS19 DATASET FILTERED PARAMETERS FOR THE SELECTED EXAMPLE

Attributes	Value
Urban/rural	Urban
Carriageway hazard	None
1st road class	Motorway
Speed limit	70 mph
Road type	Dual carriageway
2nd road class	Out of range
Road surface condition	Dry
Junction location	Not within 20 m
Junction detail	N/A
Junction control	N/A
Weather condition	Fine no high winds
Lighting condition	Daylight
Vehicle maneuver	Overtaking moving vehicle

```
# Dynamic elements
INITIAL: Ego [V1] in map location [Lane 1] of [Road 1] AND on-road
vehicle [V2] in map location [Lane 2] of [Road 1] at relative [RSR]
location to [V1]
WHEN: Ego [V1] is [Going ahead] in map location [Lane 1] of [Road 1]
DO: On-road vehicle [V2] maneuver as:
  Phase 1: [Drive] and [Moving towards] with relative [Acceleration]
           speed and [RSR] location to [V1]
           WHILE: [V2] relative location to [V1] is NOT [FSR]
  Phase 2: [Drive] and [Moving away] with relative [Decelerate] speed
           and [FSR] location to [V1]
           WHILE: [V2] relative locaiton to [V1] is [FSR] within a
                  margin of [Distance]
  Phase 3: [Change lane left] and [Cut-in] with relative [Decelerate]
           speed and [FSR] location to [V1]
           WHILE: [V2] relative locaiton to [V1] is NOT [F]
  Phase 4: [Drive] and [Moving Away] with relative [Constant] speed
           and [F] location to [V1]
           WHILE: [V2] relative location ot [V1] is [F]
END: Ego [V1] in map location [Lane 1] of [Road 1] OR [Collide]
with [V2]

# Scenery elements
DO: Map-roads and junctions network [Network 1] as:
Junctions: N/A
Roads:
R1: START
    Road type [Motorway] with [Straight] geometry in [Urban]
    environment
    END

# Environment elements
DO: Environment [Env 1] as:
    - Wind [Low]
    - Clouds [Low]
    - Precipitation [None]
    - Visibility [Good]
    - Time [Daytime]
```

Fig. 9. SDL level 1 description of the selected accident scenario

The next step is to match these parameters with the SDL parameters, for those where a direct match cannot be established, assumptions are made. The environment parameters contained in the accident data are high level description in terms of weather, lighting and time, as shown in Table IV. For the dynamic element, it is assumed that the on-road vehicle is performing an overtaking of the ego vehicle, upon overtaking, the on-road vehicle changes into ego vehicle's lane and starts decelerating, while the ego vehicle keeps going ahead without braking and results into an accident.

The SDL descriptions (Fig. 9 and Fig. 10) illustrate the complete scenario of the selected STATS19 accident example. Unlike the dynamic element and environment element, the executability of the scenery element is difficulty to examine due to the limited software available that can achieve auto-generation of the simulation scene. However, SDL can provide an understandable and common format for describing the scenery and environment element which complies with the standards defined by standards organizations.

B. Case Study 2: Formating Euro NCAP scenarios in SDL

The European New Car Assessment Programme (Euro NCAP) is a European car safety performance assessment program. It contains a series of vehicle tests, which represent major real-life accident scenarios that could result in injuries or casualties. Among the accident scenarios, the car-to-car rear

```
# Dynamic elements
INITIAL: Ego [V1] in [R1.L1] AND on-road vehicle [V2] in [R1.L2] at
[RSR] relative position
WHEN: Ego [V1] is [Going ahead] in [R1.L1]
DO: On-road vehicle [V2] maneuver as:
  Phase 1: [V2] Drive_Towards [-, 60 to 70, 2 to 3] [V1: 5 to 15, RSR]
  Phase 2: [V2] Drive_Away [-, 55 to 65, -1 to -2] [V1: 0 to 10, FSR]
           WHILE: [V2] relative locaiton to [V1] is [Within] a
                  [Longitudinal] margin of [6.5 to 7.5]
  Phase 3: [V2] LaneChgLeft_CutIn [-, 55 to 65, -1 to 1] [V1: 0 to 10,
           FSR]
  Phase 4: [V2] Drive_Away [-, 45 to 55, -0.5 to 0.5] [V1: -10 to 0, F]
END: [V2] in [R1.L1] at [F] relative position to [V1] [Not within] a
     [Longitudinal] margin of [80 to 100] OR [V1] [Collide] with [V2]

# Scenery elements
DO: Map-roads and junctions network [Network 1] as:
Junctions: N/A
Roads:
R1:START
    Road type [Motorway] as [R1] with zone as [N/A] AND speed limit
    of [70] in [Urban] environment
    - Number of lanes [3] as [R1.L1, R1.L2, R1.L3]
    - Road traffic direction [Left] with lane type [Traffic lane]
    - Lane markings [Broken lines]
    - Road surface type [Uniform] with surface condition [Dry] AND
      surface feature [N/A]
    - Horizontal road geometry [Straight] with curvature radius of [N/A]
    - Vertical road geometry [Level plane]
    - Transverse road geometry [Undvided] with [No] roadside feature
    - Roadway edge feature [Line markers]
    - Length [200 to 400] AND lane width [4 to 4.2]
    END

# Environment elements
DO: Environment [Env 1] as:
      - Wind [0 to 5]
      - Clouds [0 to 1]
      - Particulates [None]
      - Precipitation [None]
      - Time [6:00 to 7:00]
      - Illumination [Daylight] with [Sun] as light source at [30 to 35]
        elevation angle AND [FSR] position # Ego vehicle initial position
      - Connectivity [None]
```

Fig. 10. SDL level 2 description of the selected accident scenario

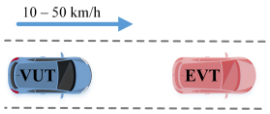

Fig. 11. Initial scene of the Euro NCAP CCRs test scenario

impact is one of the most frequent categories, this case study will present the test scenarios for the autonomous emergency braking system and convert them into SDL format. Fig. 11 displays the initial scene of car-to-car rear stationary (CCRs) test scenario defined by the Euro NCAP. During this scenario, the vehicle under test (VUT) is travelling forward at a speed range of 10-50 km/h towards a stationary Euro NCAP vehicle target (EVT). The goal is to test whether the autonomous emergency braking (AEB) is effective at reducing the VUT speed when potential collision is detected. The test shall be conducted in dry condition with no precipitation, wind speed shall be below 10m/s and natural illumination must be homogenous. Fig. 12 and Fig. 13 display the SDL description of this scenario.

```
# Dynamic elements
INITIAL: Ego [VUT] in map location [Lane 1] of [Road 1] AND
on-road vehicle [EVT] in map location [Lane 1] of [Road 1] at relative
[F] location to [VUT]
WHEN: Ego [VUT] is [Going ahead] in map location [Lane 1] of [Road 1]
DO: On-road vehicle [EVT] maneuver as:
  Phase 1: [Stop] and [Away] with relative [Constant] speed and [F] location
           to [VUT]
END: [VUT] [Stop] in map location [Lane 1] of [Road 1] OR [VUT] [Collide]
     with [EVT]

# Scenery elements
DO: Map-roads and junctions network [Network 1] as:
Junctions: N/A
Roads:
R1: START
    Road type [Motorway] with [Straight] geometry in [Urban] environment
    END

# Environment elements
DO: Environment [Env 1] as:
      - Wind [Low]
      - Clouds [High]
      - Precipitation [None]
      - Visibility [Good]
      - Time [Daytime]
```

Fig. 12. SDL level 1 description of the Euro NCAP CCRs test scenario

```
# Dynamic elements
INITIAL: Ego [VUT] in [R1.L1] AND on-road vehicle [EVT] in [R1.L1]
at [F] relative position
WHEN: Ego [VUT] is [Going ahead] in [R1.L1]
DO: On-road vehicle [EVT] maneuver as:
  Phase 1: [EVT] Stop_Away [-, 0, 0] [VUT: -10 to -50, F]
END: [VUT] [Stop] in [R1.L1] OR [VUT] [Collide] with [EVT]

# Scenery elements
DO: Map-roads and junctions network [Network 1] as:
Junctions: N/A
Roads:
R1: START
    Road type [Motorway] as [R1] with zone as [N/A] AND speed limit of
    [70] in [Urban] environment
    - Number of lanes [3] as [R1.L1, R1.L2, R1.L3]
    - Road traffic direction [N/A] with lane type [Traffic lane]
    - Lane markings [Broken lines]
    - Road surface type [Uniform] with surface condition [Dry] AND
      surface feature [N/A]
    - Horizontal road geometry [Straight] with curvature radius of [N/A]
    - Vertical road geometry [Level plane]
    - Transverse road geometry [Undivided] with [No] roadside feature
    - Roadway edge feature [Line markers]
    - Length [200 to 400] AND lane width [4 to 4.2]
    END

# Environment elements
DO: Environment [Env 1] as:
      - Wind [0 to 5]
      - Clouds [0 to 1]
      - Particulates [None]
      - Precipitation [None]
      - Time [11:00 to 13:00]
      - Illumination [Daylight] with [Sun] as light source at [85 to 90]
        elevation angle AND [FSR] position # Ego vehicle initial position
      - Connectivity [None]
```

Fig. 13. SDL level 2 description of the Euro NCAP CCRs test scenario

V. Conclusion

This paper presents a scenario definition language concept based on two different levels of abstraction. Level 1 caters to the needs of a textual language for end users such as regulators, it

forms the functional level scenarios. Level 2 is a formalised scenario specification for ingestion by toolchains for simulation based testing or real-world testing, this represents the logical and concrete level scenarios. Conversion between the two abstraction levels is achieved by adding additional details or abstracting. Both levels include scenery, dynamic and environment aspects. The elements included in each aspect align with the six-layer model for describing scenarios, and they are referenced to the ODD taxonomy for ADSs. This paper further illustrates the process of representing accident data and testing scenario using SDL.

ACKNOWLEDGMENT

This work is supported by Streetwise and OmniCAV projects, partly funded by the Centre for Connected and Autonomous Vehicles, delivered in partnership with Innovate UK. (Grant No. 103700 and 104529 respectively). The authors will like to thank the project partners for their inputs, and the WMG centre of High Value Manufacturing (HVM) Catapult and WMG, University of Warwick for providing the necessary infrastructure for conducting this study. WMG hosts one of the seven centres that comprise the HVM Catapult in the UK.

REFERENCES

[1] B. P. Derler, M. Ieee, E. A. Lee, F. Ieee, A. S. Vincentelli, and F. Ieee, "Modeling Cyber – Physical Systems," vol. 100, no. 1, 2012.

[2] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles : opportunities , barriers and policy recommendations," Transp. Res. Part A, vol. 77, pp. 167–181, 2015.

[3] D. J. Fagnant and K. M. Kockelman, "The travel and environmental implications of shared autonomous vehicles , using agent-based model scenarios," Transp. Res. Part C Emerg. Technol., vol. 40, pp. 1–13, 2014.

[4] S. Le Vine, X. Liu, F. Zheng, and J. Polak, "Automated cars : Queue discharge at signalized intersections with ' Assured-Clear-Distance-Ahead ' driving strategies," Transp. Res. Part C Emerg. Technol., vol. 62, pp. 35–54, 2016.

[5] N. Balfe, S. Sharples, and J. R. Wilson, "Impact of automation : Measurement of performance , workload and behaviour in a complex control environment," Appl. Ergon., vol. 47, pp. 52–64, 2015.

[6] R. N. Charette, "This Car Runs on Code," IEEE Spectrum, Feb-2009.

[7] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings, "Identifying a gap in existing validation methodologies for intelligent automotive systems: Introducing the 3xD simulator," in Proc. of the IEEE Intelligent Vehicles Symposium 2015, 2015, pp. 648–653.

[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," SysCon 2019 - 13th Annu. IEEE Int. Syst. Conf. Proc., 2019.

[9] 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.

[10] 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.

[11] Euro NCAP, "Euro NCAP 2025 Roadmap: In pursuit of Vision Zero," 2017.

[12] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings, "The Science of Testing: An Automotive Perspective," in SAE Technical Papers, 2018, vol. 2018-April.

[13] ASAM e.V., "ASAM OpenSCENARIO," 2020.

[14] W. C. Information, "WISE Drive : Requirements Analysis Framework for Automated Driving Systems," 2020. .

[15] 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.

[16] K. Go and J. Carroll, "The blind men and the elephant: views of scenario-based system design," Interactions, vol. 11, no. 6, pp. 44–53, 2004.

[17] S. Geyer et al., "Concept and development of a unified ontology for generating test and use-case catalogues for assisted and automated vehicle guidance," IET Intell. Transp. Syst., vol. 8, no. 3, pp. 183–189, 2014.

[18] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and Substantiating the Terms Scene , Situation , and Scenario for Automated Driving," in Proc. of the 2015 IEEE 18th 18th International Conference on Intelligent Transportation Systems, 2015.

[19] E. de Gelder et al., "Ontology for Scenarios for the Assessment of Automated Vehicles," 2020.

[20] E. De Gelder and J. P. Paardekooper, "Assessment of Automated Driving Systems using real-life scenarios," IEEE Intell. Veh. Symp. Proc., no. Iv, pp. 589–594, 2017.

[21] A. Pütz, A. Zlocki, J. Bock, and L. Eckstein, "System validation of highly automated vehicles with a database of relevant traffic scenarios," 12th ITS Eur. Congr., no. June, pp. 1–8, 2017.

[22] 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.

[23] J. Bock, R. Krajewski, L. Eckstein, J. Klimke, J. Sauerbier, and A. Zlocki, "Data basis for scenario-based validation of HAD on highways," in 27th Aachen Colloquium Automobile and Engine Technology, 2018.

[24] H. Grönninger, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel, "Textbased Modeling," Proc. 4th Int. Work. Softw. Lang. Eng. (ateM 2007), no. 4, 2007.

[25] B. D. Cruz, B. Jayaraman, A. Dwarakanath, and C. McMillan, "Detecting Vague Words & Phrases in Requirements Documents in a Multilingual Environment," Proc. - 2017 IEEE 25th Int. Requir. Eng. Conf. RE 2017, pp. 233–242, 2017.

[26] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO--simulation of urban mobility: an overview," Proc. SIMUL 2011, Third Int. Conf. Adv. Syst. Simul., 2011.

[27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," no. CoRL, pp. 1–16, 2017.

[28] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps. Pervasive Computing," IEEE Pervasive Comput., vol. 7, no. 4, pp. 12–18, 2008.

[29] M. Dupuis, M. Strobl, and H. Grezlikowski, "OpenDRIVE 2010 and Beyond – Status and Future of the de facto Standard for the Description of Road Networks," Driv. Simul. Conf. 2010 Eur., pp. 231–242, 2012.

[30] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," IEEE Intell. Veh. Symp. Proc., no. Iv, pp. 420–425, 2014.

[31] M. Althoff, S. Urban, and M. Koschi, "Automatic Conversion of Road Networks from OpenDRIVE to Lanelets," Proc. 2018 IEEE Int. Conf. Serv. Oper. Logist. Informatics, SOLI 2018, pp. 157–162, 2018.

[32] C. Pilz, G. Steinbauer, M. Schratter, and D. Watzenig, "Development of a scenario simulation platform to support autonomous driving verification," 2019 8th IEEE Int. Conf. Connect. Veh. Expo, ICCVE 2019 - Proc., 2019.

[33] R. Queiroz, T. Berger, and K. Czarnecki, "GeoScenario: An open DSL for autonomous driving scenario representation," IEEE Intell. Veh. Symp. Proc., vol. 2019-June, no. Iv, pp. 287–294, 2019.

[34] SAE International, "Surface Vehicle Recommended Practice - J3016," 2018.

[35] "Road vehicles — Taxonomy for operational design domain for automated driving systems," Int. Organ. Stand. [ISO], 2020.

[36] "Road Safety Data - STATS19," UK Department for Transport, 2020.