MDPI

# Traffic Scenarios for Automated Vehicle Testing: A Review of Description Languages and Systems

**Jing Ma** [1,2] [ID]**, Xiaobo Che** [1] [ID]**, Yanqiang Li** [1,3,*] [ID] **and Edmund M-K Lai** [2] [ID]

1. Institute of Automation, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China; jing.ma@aut.ac.nz (J.M.); chexb@sdas.org (X.C.)
2. School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand; edmund.lai@aut.ac.nz
3. School of Control Science and Engineering (CSE) of Shandong University, Jinan 250014, China
* Correspondence: liyq@sdas.org

**Abstract:** Testing and validation of the functionalities and safety of automated vehicles shifted from a distance-based to a scenario-based method in the past decade. A number of domain-specific languages and systems were developed to support scenario-based testing. The aim of this paper is to review and compare the features and characteristics of the major scenario description languages and systems (SDLS). Each of them is designed for different purposes and with different goals; therefore, they have their strengths and weaknesses. Their characteristics are highlighted with an example nontrivial traffic scenario that we designed. We also discuss some directions for further development and research of these SDLS.

check for updates

## 1. Introduction

The development of automated vehicles (AV) progressed rapidly in the past few years. Many companies were testing their AVs on roads and highways in the United States and other nations around the world [1,2]. However, the roll-out of AVs on public roads took longer than expected [3], and one of the main reasons is related to trust. Can the public trust that these AVs are safe? The answer to this question is complicated by the fact that an AV integrates both the mechanics of a vehicle and the ability of the driver into a single entity. Any safety testing regime has to include both the mechanics of the car and its ability to drive, which currently is handled by driver licensing.

One way to ensure the safety of AVs is to have it driven enough. How much distance an AV has to be driven to adequately evaluate its safety was hotly debated. It was argued that to make statistical safety comparisons with the performance of human drivers, hundreds of millions of miles of driving is required [4]. However, doing so would take decades and is obviously impractical. Therefore, the focus of AV testing shifted from a driving distance-based approach to a scenario-based approach [5–7]. This later approach also aligns well with the development process described in the 2016 version of the ISO-26262 standard [8] which is concerned with the development of safety-critical electrical and electronic systems.

Scenario-based testing requires well-designed traffic scenarios that can sufficiently test the capabilities of the automated driving systems [7]. A driving scenario is typically made up of static, dynamic, and temporary elements. These include the environment such as road widths, number of lanes, road curvature, road signs, pedestrian sidewalks, a set of vehicles (automated or human-driven) and their initial states (position, orientation, speed, etc.), pedestrians, traffic signals, and the ego automated vehicle (the AV under test) with its initial state and ultimate goal. It can also be viewed as a time sequence of scenes [5,9]. Such

scenarios need to be described and specified, either textually or graphically or both. There are now a variety of scenario description languages and systems (SDLS) which are either simulation platform dependent or independent. Choosing the right SDLS is important to researchers and developers of AVs, as well as regulatory authorities who are concerned with safety testing.

In this paper, we provide a review of the characteristics and capabilities of a wide range of SDLS. Firstly, Section 2 briefly describes what a driving scenario constitutes. This is followed by descriptions of various SDLS in Section 3. We make use of a specific scenario to illustrate their capabilities and characteristics. A discussion on the relative merits of these SDLS is presented in Section 4. Finally, Section 5 concludes with some recommendations.

## 2. Driving Scenarios

In the context of AV testing, there is no single agreed-upon definition of a traffic scenario. Several different definitions could be found in the literature. In [10], Go et al.consider a scenario as a description that contains actors, background of actors, environment and sequences of actions. Along the same line, Geyer et al. [11] think that a scenario should include both the scenery and the dynamic elements of a scene. The definition in [12] focuses on the temporal development in a sequence of scenes. With this approach, each scenario starts with an initial scene. Action and events as well as goals and values may be specified to characterize the temporal evolution of the scenario.

Elrofai et al. [13] considers time duration an important element of a scenario specification. They describe a scenario as a typical maneuvre on the road with a complete set of relevant conditions and trajectories of other traffic participants that interacts with the ego vehicle over a specified duration, typically in the order of seconds. To Gelder et al. [14], the focus of the scenario should be on the ego vehicle. Thus, a scenario is a quantitative description of the relevant characteristics of the ego vehicle, its activities and/or goals, its static environment, and its dynamic environment. In addition, a scenario also contains all events that are relevant to the ego vehicle.

Given the variety of details that could be included in a scenario description, it may be necessary to organize the information in a way that promotes clarity and efficiency. A two-level approach is proposed in [15]. Level 1 is a higher-level functional description which is text-based, suitable for use by developers and regulators. Level 2 is in a machine-readable form that can be imported by simulators and other test systems.

There are also suggestions to divide the vast amount of details into groups. A five-layer model was proposed in [16,17]. It is later further extended to include a sixth layer [18]. In this scheme, the first layer describes the road layout, including the basic geometry of the streets with surface boundaries. Traffic signs and barriers are in the second layer. Any temporal aspects of the first two layers are described in the third layer. This is followed by information on the moving and stationary objects in layer 4, and environmental information such as weather in layer 5. The final layer is for all aspects of data connectivity and communications.

## 3. Scenario Description Languages and Systems

A number of SDLS were developed by academia and by industry. Some are tightly integrated with their development frameworks, while others are open and can be exported to various simulators. Some of them are text-based and others are graphical. They are designed for different kinds of end-users with different requirements.

### 3.1. Example Scenario

To compare and illustrate the capabilities and characteristics of various SDLS in subsequent sections, we shall make use of a nontrivial scenario that is illustrated in Figure 1. This example scenario is as follows: it is a daytime scenario with the ego AV on a minor road. The road surface is dry and the weather is fine without high winds. The goal is for the ego AV to turn left at the T-junction onto the main road. There is a "Give Way" sign at

the corner of the minor road. The minor road has one lane in each direction while the main road has two lanes in each direction; all lanes are four meters wide. On the main road, there are five vehicles, V1, V2, V3, V4, and V5, moving in the directions shown. There are also three parked vehicles, V6, V7, and V8, on the main road.
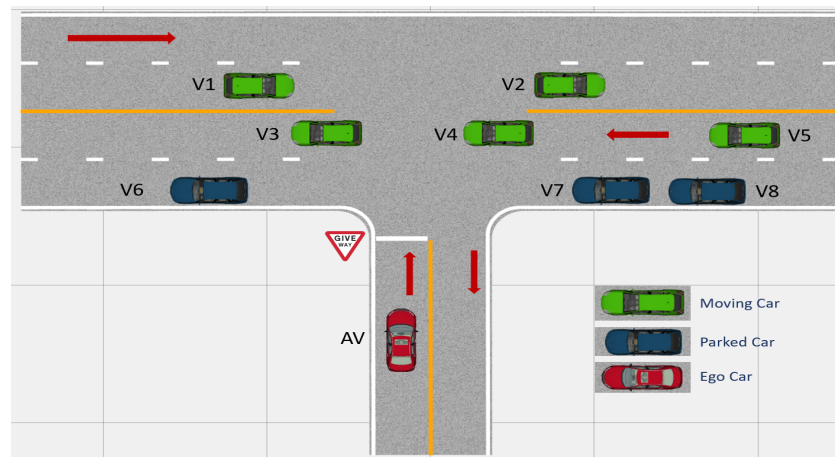


**Figure 1.** Test scenario for comparison.

This scenario is more complex than most of the highway, round-about, and lane-changing scenarios used in a lot of research publications. It consists of more than one lane on the main road in each direction. This could be used as an edge scenario for testing as the parked cars V7 and V8 may prevent the sensors on the AV to "see" V5. There is also a traffic sign that the AV need to observe. These elements are used to illustrate the capabilities or the deficiencies of the SDLS reviewed.

*3.2. stiEF*

The description of the scenario given in Section 3.1 is probably how a scenario designer would have written as a first step. The main concerns with natural language descriptions are ambiguity and contradiction. These concerns will need to be removed so that the scenario to be formalized and be able to be exported to other computer tools. Also, all specific details have to be specified [15]. For example, an obvious detail that is missing in our example is the speed and any acceleration or deceleration of the moving vehicles.

There are attempts to use natural language for such descriptions. For example, RE-CAA [19] is introduced as an automated requirements engineering process which extensively uses natural language parsing technologies to analyze plain textual requirements and remove ambiguities and errors. It is generally assumed that the requirements are described in English. The above process will need to be repeated for every translation of the English text.

The domain-specific description language *stiEF* is designed to prevent misunderstandings in a multilingual setting. It defines a formal grammar for specifying scenarios based on multiple languages. *stiEF* stands for scenario-accompanied, text-based, iterative evaluation of automated driving function. It was developed by the Car.Software Organization, now renamed as CARIAD, in 2019 [20].

The most significant feature of the language defined by *stiEF* is that it resembles natural languages. Hence, the descriptions can be understood by humans easily. Furthermore, it eliminates the chance of having translation errors between the natural languages that it supports. At the moment, it supports English and German. The idea is that, over time, more natural languages will be added, for instance Chinese. Figure 2 shows a small part of a scenario described in both English and German. It illustrates how a description can be expressed both languages. Lines 2 and 3 are equivalent to each other.

*stiEF* also provides tools to import scenarios written in free-form natural languages. The imported descriptions are parsed and iteratively refined until they are fully translated

into the *stiEF* language. In addition, development tools support the instant visualization of the scenarios as they are being written in the language of *stiEF*. Scenario descriptions can also be exported in JSON and XML formats. The later format allows openSCENARIO and openDRIVE (to be discussed in the Section 3.3) to import the scenario descriptions and provide integration with simulation tools such as Vires $VTD^2$ [21].

Another advantage of *stiEF* is that scenario descriptions are based on the 5-layer model, as described in Section 2. Figure 3 shows a 5-layer description using the constrained natural language of *stiEF*.

While the language defined by *stiEF* resembles natural languages, it is very restricted. Therefore, the quality of the scenario descriptions depends very much on the quality of the language. Additionally, *stiEF* scenarios are currently only limited to freeways and highways. Hence, it is unable to express our example scenario in Section 3.1.



**Figure 2.** Bilingual characteristics of *stiEF*.
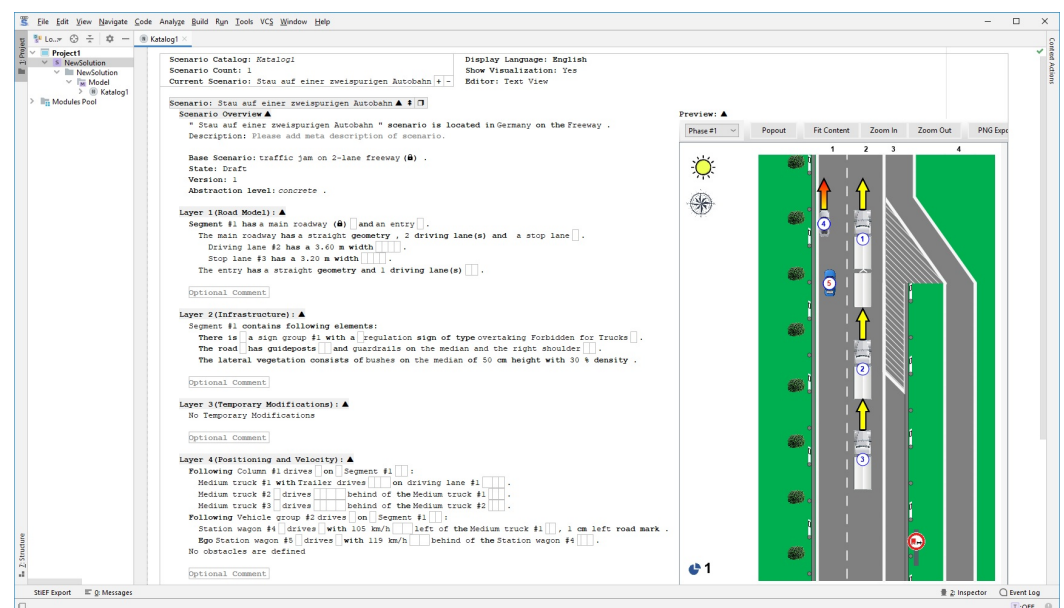


**Figure 3.** Example of scenario description using *stiEF*. From [20].

*3.3. OpenSCENARIO*

OpenSCENARIO [22] aims to be an open standard for the description of the dynamic aspects of driving scenarios. It was initially developed by VIRES Simulationstechnologie GmbH, a German company that provides transportation simulation solutions. In 2018, the ownership and future development of OpenSCENARIO was transferred to the Association for Standardization of Automation and Measuring Systems (ASAM) which is a standardization organization formed by some of the major European car manufacturers in 1998. It defines an open file format for the description of a scenario and is part of a suite of three open standards that includes OpenDRIVE and OpenCRG.

In OpenSCENARIO, every scenario is completely described by the Storyboard. It contains the timeline of events that occur within that scenario. This includes all the actions and maneuvers and the conditions that trigger them. It describes entities such as vehicles and pedestrians. It can also reference entities that are described in other file formats. For example, a "RoadNetwork" contains a reference to the static driving infrastructure

described in the openDRIVE format. This allows openSCENARIO to refer to entities such as lanes. The road network description could also include reference to external files that describe the environment and may contain 3D models.

The OpenSCENARIO data model is defined by the Unified Modeling Language (UML) [23]. UML enhances readability by graphical data model diagrams, and was widely adopted by the software engineering industry for object-oriented modeling and design. While UML supports use-case diagrams, class diagrams, and activity diagrams, only class diagrams are used for data modeling in OpenSCENARIO. XML schemes can be derived from these data models and used for the verification of the XML data files.

Figure 4 shows part of the openSCENARIO XML file that describes the example scenario of Section 3.1. It shows one of the stories in the Storyboard that involves a maneuver of the ego vehicle to turn left.

```
<OpenSCENARIO>
    <RoadNetwork>
        <LogicFile filepath="xxx.xodr"> // OpenDRIVE File
    </RoadNetwork>
    <Entities>
      <ScenarioObject name = "Ego Vehicle AV">
      <ScenarioObject name ="V1">

      ....
    </Entities>
    <Storyboard>
      <Init> </Init>  // Locate entities using World Position
        <Story name ="Turn left onto the Main Road">
          <Act name ="Act1">
            <Maneuver name ="TurnLeftManeuver">
              <Event name ="TurnLeftEvent" priotiry="overwrite">
                <Action name = "TurnLeftAction">
              </Event>
            </Maneuver>
          </Act>
        </Story>
    </Storyboard>
</OpenSCENARIO>
```

**Figure 4.** OpenSCENARIO description of part of example scenario.

While openDRIVE describes the road network including its geometry, lanes and other objects such as traffic signals, openCRG [24] is used to describe the 3D geometries such as inclination, pitch, and yaw angles of the roads. These three open-source standards–OpenSCENARIO, OpenDRIVE, and OpenCRG complement each other to cover all the static and dynamic contents required for scenario description and simulation. The XML files can be imported by different simulation platforms such as $ViresVTD^2$ [21], Matlab [25], CARLA [26] and PreScan [27].

### 3.4. Hesperia

*Hesperia* is a software framework that was designed for a research and development project on AV testing called "CarOLO" in 2007 [28,29]. This project was carried out at the University of California, Berkeley in collaboration with RWTH Aachen University. In this project, the autonomous vehicle named "Caroline" was developed for the 2007 DARPA Urban Challenge. *Hesperia* is a component-based framework for scenario-based development. A scenario is described by a story card that includes the title, date and location. It can also include an image depicting the road layout and surrounding environment. More details of the functional requirements are provided in tabular form at the bottom of the card. An example of a story card is shown in Figure 5a.

Based on the story card, precise description of a scenario is entered using a graphical editor. This graphical editor is based on the Eclipse Rich Client Framework and reuses the Java sources from the MontiCore framework. The information of the scenario is encoded by a domain specific language (DSL), which allows a hierarchical representation. The description could include abstract elements and complex models in 3D. A bird's eye view of the scenario layout is shown in Figure 5b. Elements can be added to the scenarios by simply using drag-and-drop.

A part of our example T-junction scenario is coded using the *Hesperia* DSL and shown in Figure 6. This is a description at the highest level. Lower levels of descriptions contained in other files are referenced.
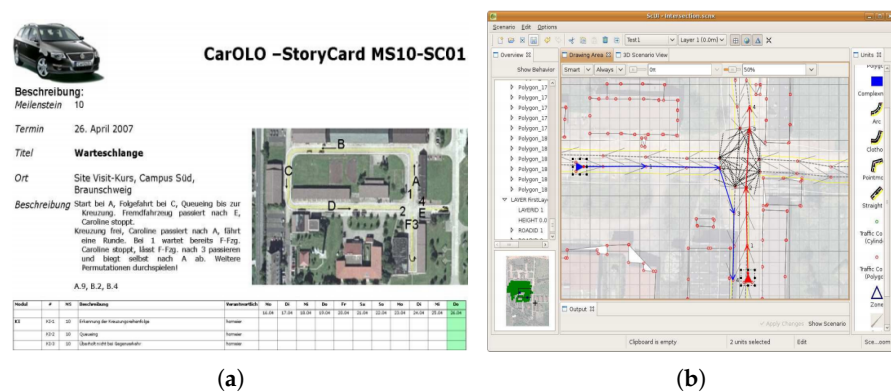


(a)                                                   (b)

**Figure 5.** Hesperia framework scenario description [29]. (**a**) Story Card by project CarOLO. (**b**) Graphical Editor for using DSL.

```
ScenarioFile =

    ScenarioHeader // File header.
        Ground // Ground layer.
        Layer+ ";"; // Data layers containing roads and other stuff.


ComplexModel =
    T-junction:"ComplexModel"
    "ModelFile" ModelFile: Ego AV
    "Position" Position: At the minor road before T-junction


SituationHeader =
    "Situation" SituationName:Turn left onto the main road
```

**Figure 6.** Hesperia description of T-junction scenario.

### 3.5. SceML

SceML is a graph-based scenario modeling language that was developed by FZI Research Center for Information Technology [30]. The goal of this hierarchical graphical scenario language is to help engineers encode scenarios in a machine readable data structure. It follows the UML class diagram conventions and supports different abstraction levels of scenario description. The main feature of SceML is that the graph-based structure provides modularity and reusable subscenarios.

It is also able to generate and extract scenarios from recorded data sets with machine learning algorithms or created by experts. The modeling language allows different depth

of information modeling to support different abstraction level during the development process. SceML utilized sequential and parallel executions policies.

Each scenario in SceML is defined by a behavior tree. The tree starts with a root node that contains its name, action lists, and other parameters. The children nodes are comprised of maneuver and condition nodes, join nodes, and modules. Maneuver nodes show movements while condition nodes help to synchronize maneuvers. Join nodes symbolize the parallel execution of nodes or sequences. Modules allow to summarize maneuvers and conditions, which can be save in a library and reused in further scenarios. Finally, each scenario terminates at an end node where all maneuver paths converge.

Figure 7 illustrates the turn left onto main road maneuver in our T-junction example scenario in this graph notation. As discussed above, each scenario has one root node that contains a list of all the actors in the scenario, which are the ego AV, and other vehicles V1 to V8. The top "InLocationRadius" node in the figure depicts Ego AV reaching junction. As the Ego AV approaches the junction, it detects the distances to V3 and V4, depicted by the bottom "InLocationRadius" node. When the ego AV has enough distance between V3 and V4, it turns left onto the main road. The middle "InLocationRadius" node depicts the state after the AV has entered the main road. The white nodes are synchronization condition nodes. Ultimately, all maneuver paths converge to the end node.
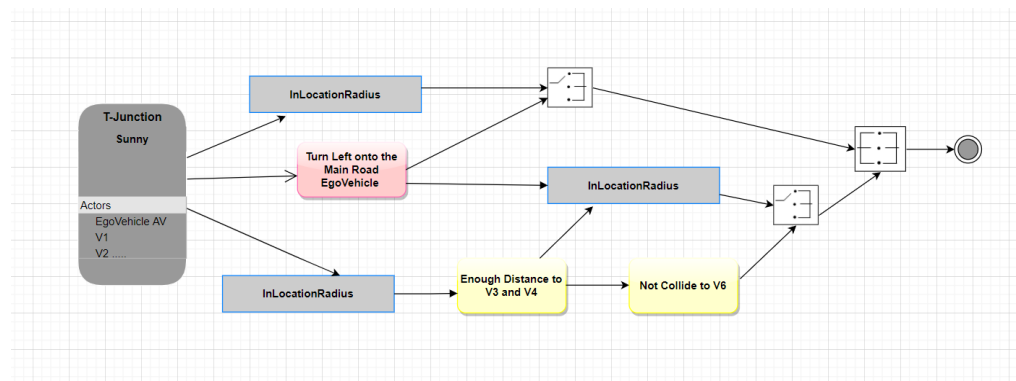


**Figure 7.** Turn left onto main road maneuver of example scenario using SceML Representation.

The graph structure in SceML can be parsed into an OpenSCENARIO XML file for model checking. The scenario may then be imported into a simulator such as CARLA [30]. Thus SceML can be used to design concrete scenarios which could then be validated. One characteristic of SceML that makes it different from other SDLS is that the scenario outcome does not need to be defined before execution. Therefore, many variations of the basic scenarios could potentially be combined. It could also be used to inspect and modify automatically generated and extracted scenarios from recorded data sets.

*3.6. OAS*

Open Autonomous Safety (OAS) [31], developed by Cruise LLC, is a scenario description framework where the scenario is divided into behavioral stages. Each stage of a scenario consists of a combination of a 2D view of the layout, a natural language description of the scenario, and the expected result. The layout includes the states of all actors in space and the actors' behaviors in that stage. The expected result is needed to determine if the ego AV achieves the objective. Only when the expected result is achieved will the AV be allowed to proceed to the next stage.

In the OAS system, there is a unique ID for each scenario. Each ID has to follow a strict format—"road segments-lanes-stop signs-scenario category-Ego action-other actors". So, our example scenario would be given an ID "2-2-NW-TL-L-CAR:Pa:02". Following the requirements of OAS, our example scenario could be divided into three behavioral stages, as shown in Figure 8. Each stage represents a safety critical section of the scenarios

that requires testing to ensure that ego vehicle AV can safely navigate its surrounding environment.
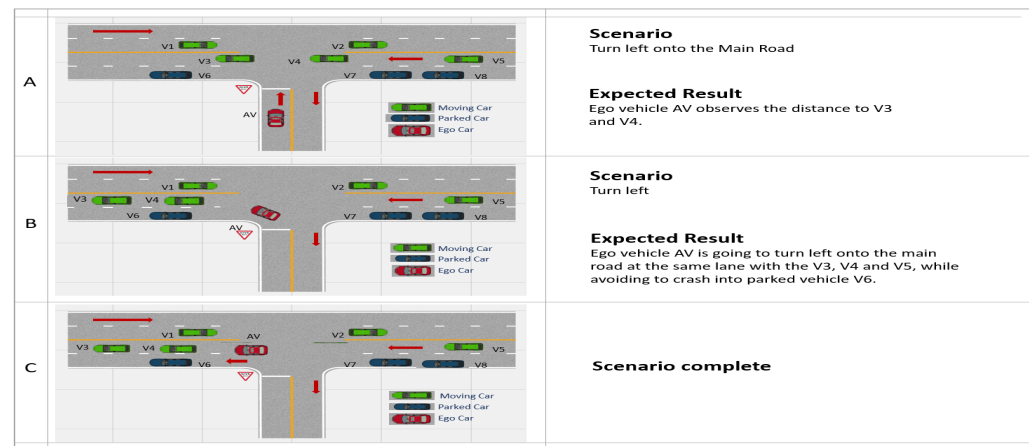


**Figure 8.** Turn left onto main road using Open Autonomous Safety.

The complete OAS scenario consists of both textual and graphical descriptions. The graphical image shows where the actors are placed in space and the textual description using natural language describe how the actors behave. The stages clearly specify the sequence of critical events in the scenario. The complexity of a scenario depends on the number of stages. However, there is no unique way to determine the stages of a scenario. Given the same scenario, different designers may come up with different stages for testing. This makes OAS scenarios more difficult to reuse. Also, some important details such as the exact position of each actor and their status are still missing from the description. Furthermore, tools for exporting OAS scenarios to other test tools such as simulators are still under development.

*3.7. GeoScenario*

GeoScenario is an open domain specific language for scenario description in testing driving simulation [32]. Its development is managed by the Association for Standardization of Automation and Measuring Systems (ASAM) and it is in its early stage of development. The goal is to develop a language to capture test scenarios that cover the complexity of road traffic situations and promote the reuse of test cases. The language is also simple enough to be human readable, and can be easily extended with new features and specializations of its standard components.

This language aims to describe dynamic contents that include driver behavior, traffic, weather, environmental events and other features. The language is XML-based, and the static content is built on top of the Open Street Map (OSM) standard [33,34]. OSM is a well-known collaborative project to create and publish free maps using an open XML format. Lanelets [35] is an open extension of the OSM format that specially supports road network representation for both geometrical and topological aspects. GeoScenario therefore uses Lanelets to represent the Road Network of a scenario. The road network is stored in a separated XML file to make replacement easy.

Although GeoScenario and OpenScenario are managed by the same organization— ASAM, with similar goals—they differ in their structure and level of abstraction. There are two OSM primitive types in GeoScenario–Node and Way. A node is a core element of GeoScenario, representing a specific point on Earth's surface. Each node has an ID number and a pair of coordinates. Vehicles and pedestrians can also be represented by nodes. A way is an ordered list of nodes joined by straight lines. Such polyline structures can be used to define paths and boundaries. The attributes of nodes and ways can be specified using tags.

A GeoScenario language is system independent. It can represent a diverse range of traffic situations that could be manually designed by experts, extracted from real traffic data, or imported from driving databases. Instead of defining actions and maneuvers for the AV, GeoScenario only specifies initial conditions and goals. The ego AV always starts a scenario in a parked position. Both static objects and dynamic elements are included in the scenario. Trigger nodes can be added to activate different actions over dynamic elements. Figure 9 shows both the graphical depiction of our example scenario and its corresponding XML file describing some nodes and a way.
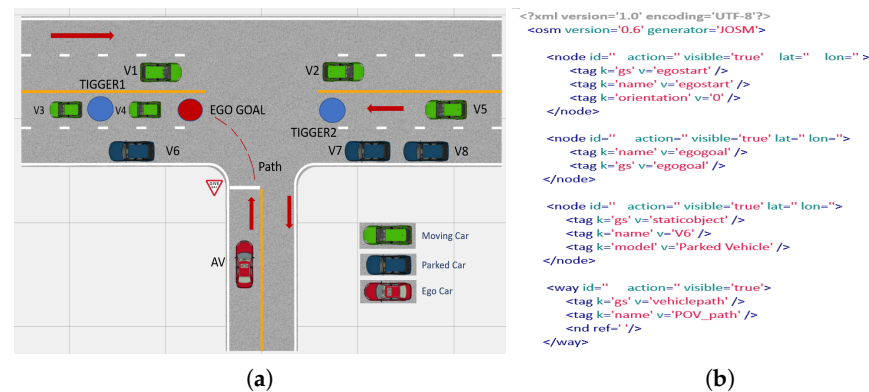


| (**a**) | (**b**) |
|---------|---------|

**Figure 9.** Defining example scenario using GeoScenario. (**a**) Graphical representation of the scenario. (**b**) Part of the corresponding GeoScenario XML File.

Java OpenStreetMap Editor (JOSM) [36] is commonly used to edit GeoScenarios files. By adding a set of custom development tools, scenarios can easily be defined on top of Lanelets and other map layers. Additional tools continue to be developed. One of them is a GeoScenario Checker. There are currently over 130 scenario-based test cases available in GeoScenarios for testing engineers.

### 3.8. CommonRoad

CommonRoad [37] is developed by a team from the Technical University of Munich in Germany. It is a platform-independent format for specific road traffic scenarios for motion planning on roads. CommonRoad scenarios are in the form of XML files, describing the the road network, static and dynamic obstacles, and the planning problem of the ego vehicle. Part of the description of our example scenario in CommonRoad XML format is shown in Figure 10.

CommonRoad uses lanelets as a road description. This allows the road network to be modeled as a directed graph. By definition, lanelets are "atomic, interconnected, drivable road segments geometrically represented by their left and right bounds", which may carry additional data to describe the static environment. CommonRoad incorporates a small subset of OSM attributes. Hence, these bounds are encoded by an array of OSM nodes forming a polyline. Together, they compose the lanelets map. Lanelets can be used to compose the road network of a scenario [32]. Not only is it able to compose complex road situations, but it can also incorporate tactical information for maneuver generation. In addition, the format of OpenDRIVE can be converted to lanelets in CommonRoad.

Motion planning involves specifying the initial state such as the position and velocity of the vehicles, and one or more goal states. The simulation tool SUMO [38] can read CommonRoad XML files and perform simulations. It is necessary to specify a time step size so that snapshots of the scenario can be encoded in CommonRoad for viewing.

```
<?xml version="1.0" encoding="utf-8"?>
<commonRoad commonRoadVersion='2020a' benchmarkID='example'
date=" author="" affiliation="" source="OpenStreetMaps (OSM)" timeStepSize='0.1'>
    <location>
        <geoNameId>   </geoNameId>
    </location>
    <scenarioTags>
        <T-junction/>
    </scenarioTags>
    <lanelet id=' '>
    ...
    </lanelet>
    <T-junction id=' '>
    ...
    </T-junction>
    ...
    <staticObstacle id=' '>
        <type>V6</type>
        <position>
    ..
    </staticObstacle>
    <dynamicObstacle id=' '>
        <type>V4</type>
        <initialState>
        ...
    </dynamicObstacle>
    <planningProblem id=' '>
        <initialState>
        <goalState>
    </planningProblem>
</commonRoad>
```

**Figure 10.** CommonRoad Scenario Example.

There are now over 2000 traffic scenarios available to download in the CommonRoad scenario database. These scenarios are generated by naturally occurring and handcrafted dangerous situations.

Lanlets2 [39] is an improvement of Lanelet to incorporate a high-definition map framework for highly automated driving. However, both Lanelets and Lanelets2 are not able to support global route planning [40]. Development of large scale lane-level road network descriptions are still in the early stages. Much further work is needed to support AV development and testing.

*3.9. SCENIC*

SCENIC is an imperative, object-oriented probabilistic programming language for describing driving scenarios. It was developed by a research team from the University of California, Berkeley in 2019 [41]. Although SCENIC is well suited for the domain of generating data for perception systems, the syntax of the language is designed to simplify the task of writing complex scenarios, and to enable the use of specialized sampling techniques. Its main goal is to design a scene system that can capture corner case scenarios based on machine learning data sets.

Probabilistic programming languages are used to describe probabilistic models and perform inference based on these models. Other such languages include PROB [42] and Church [43]. The syntax of SCENIC is largely devoted to expressing geometric relationships between objects in a concise yet readable manner. The biggest advantage of using SCENIC is to generate specialized test sets, and to improve the effectiveness of training sets by emphasizing difficult cases. There are classes, objects, geometry, distributions and coordinate systems. SCENIC has provided a small library which defines the types of

objects supported by simulators, and an interface which converts SCENIC into the input format required by such simulators. SCENIC is also able to work with simulators such as open-source VERIFAI toolkit [44], Webots [45], LGSVL [46], and CARLA [26] with 3D scenes and encoding dynamics to enable generation of videos instead of static scenes. Each individual simulator has a specialized SCENIC interface which requires additional setup. There are many SCENIC scenarios specifically tailored for these simulators. OpenDrive and OAS can be utilized to describe road networks in SCENIC.

There are also many cross-platform scenarios written in SCENIC's abstract application driving domains. These scenarios can be visualized without the need for a simulator. Figure 11 shows a small section of the description of our example scenario. It is generated by using the Webots simulator.

```
from scenic.simulators.webots.road.world import setLocalWorld
setLocalWorld(__file__, 'T-junction.wbt')

from scenic.simulators.webots.road.model import *
# * represent Scenic or Python module

ego = AV

for i in range(4):
        spot = Oriented Point on visible curb
        AV ahead of (spot offset by -0.25 @ 0)
```

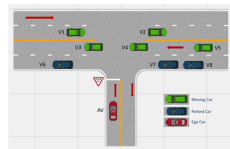**Figure 11.** Example scenario in SCENIC.

*3.10. M-SDL*

In 2019, the scenario description language M-SDL was released by Foretellix Ltd., a start-up company based in Israel [47]. The aim is to provide nonproprietary scenarios with a good combination of power, readability, and composability based on an open language. Foretellix was founded by a team with expertise in measurable verification and validation in the semiconductor industry. Their stated mission is to enable measurable safety of advanced driver assistance systems (ADAS) and AVs. Being an open language, M-SDL could be adopted by different testing platforms with dual interpretation (both active and passive). It supports several levels of abstractions and morphed for new Operational Design Domains (ODDs). Over 500 engineers from 250 different entities have already downloaded the M-SDL specification.

The syntax of M-SDL is python-like and it is mainly a declarative programming language. It is used in the OpenSCENARIO 2.0 concept document. Using M-SDL, it is easy to specify any mix of scenarios and operating conditions to identify previously unknown hazardous core or edge cases. M-SDL is very friendly to both test-writers and scenario developers. The former does not have to learn the full M-SDL, they only need to generate multiple tests building on developed scenarios. While the latter can leverage existing libraries to create more abstract scenarios. The same scenario for both left- and right-hand side traffic flow to avoid duplicating scenario creation and maintenance efforts.

There is a graphical user interface (GUI) which is named Foretify to select a map and draw a scenario in [48]. The scenario can be extended by using specifying different combinations of parameter via a spreadsheet as shown in Figure 12. A large number of meaningful core and edge cases can be generated from a single basic scenario.

| Using spreadsheets to quickly try many combinations for example scenario | | | | |
|---|---|---|---|---|
| Runs | Simulator | AV_direction | V4_direction | V5_direction |
| 10 | Carla | left | Straight_take over_ AV | Straight |
| 100 | SUMO | left | Straight | Straight |
| 20 | Carla | left | Straight | Straight_take over_ AV |
| 18 | SUMO | left | Straight_take over_ AV | Straight |



- Consider the table over, each line will run many combinations of our example scenario.
- The table or any spreadsheets can be used to generate scenarios after executing many combinations.

**Figure 12.** Using spreadsheets to try combination by M-SDL [48].

The building blocks of M-SDL are statements, structures, actors, scenarios members, scenario invocations and expressions. Since M-SDL has a python-like syntax and lexical conventions, generic abstract scenarios can be easily defined. Part of the M-SDL description of our example scenario is shown in Figure 13.

```
# The T-junction scenario

scenario t_junction:
    car1: AV    # The ego vehicle
    side: av_side # A side: left or right
    path: path

    path_min_driving_lanes(path: path, min_driving_lanes: 2) # at least two lanes

    do serial():
        av.drive(path: path, adjust: true) with:
                position(distance: [5..100]m)
        turn_left: parallel(duration: [2..5]s):    # turn_left
```

**Figure 13.** Example scenario in M-SDL.

## 4. Discussion

A number of competing and complimentary scenario description languages and systems were discussed in the previous section. Their characteristics can be summarized in Table 1. For each SDL, this table lists their primary data format for scenario description, and whether they are open-source. Most of them make use of an external standard or system for describing the road network and its conditions. Furthermore, we listed the simulation platforms which support each SDLS. This is important, as scenario descriptions are mostly used for simulation studies. If a researcher, developer, or organization is committed to a particular simulation software, then their choice of SDL is restricted. The simulator CARLA, which is an open-sourced software itself, supports the most SDLS.

Each SDLS is designed with different goals in mind. For example, openSCENARIO aims to be an open standard scenario description language that will be accepted by many users and simulation tools. On the other hand, M-SDL makes it easy for many test cases to be generated from a basic scenario. This feature will be useful and convenient for testing and validation purposes. But regardless of the design goals of the SDLS, tools must be available to allow their scenario descriptions to be imported by simulation tools and platforms to allow the AV to be tested in those situations. Although traditionally, simulations are carried out in 2D, several photorealistic 3D simulators are now available, for instance [22,26,27,46,47,49]. Thus, the trend is to enable the description of scenarios in 3D by incorporating photorealistic 3D models.

**Table 1.** Comparison of listed scenario description language.

| SDL | Data Format | Open Source | Road Description | Simulation Platform |
|---|---|---|---|---|
| *stiEF* | XML | Yes | OpenDRIVE | *ViresVTD*$^2$ |
| *Hesperia* | Programming language | Unknown | 3D | CxxTest Prescan |
| OAS | HTML | Yes | 2D | OAS |
| SceML | XML | Yes | OpenDRIVE | CARLA |
| OpenSCENARIO | XML | Yes | OpenDRIVE | Prescan CARLA *ViresVTD*$^2$ |
| GeoScenario | XML | Yes | Lanlets | Unreal Engine |
| CommonRoad | XML | Yes | Lanlets | SUMO |
| SCENIC | Probabilistic programming language | Yes | OpenDRIVE OSM | VERIFAI CARLA Webots LGSVL |
| M-DSL | Python-like Syntax | Yes | OpenDRIVE | CARLA SUMO |

Not only do 3D models look more realistic, the scenarios are more realistic in the sense that not all information are available to the ego AV at any given time. For instance, in our example scenario, the oncoming car V5 on the main road is obscured by the parked cars V7 and V8. It would were much easier for the ego AV to decide on its action if the position and speed of V5 is made available. However, if the sensor on the ego AV cannot pick out V5 until V5 appears in its line-of-sight, then the situation will become much more realistic. In fact, this can become one of the edge or corner cases for testing purposes.

Using text or graphics alone may be sufficient describe simple scenarios. However, as pointed out earlier, simple scenarios are unlikely to be sufficient for testing and verification of safety purposes. Therefore, a combination of textual and graphical representations will be needed, especially when real-world maps and weather data are to be incorporated.

Currently, scenarios are almost entirely designed by human experts. This approach greatly limits the number of test scenarios that can be generated. Combing data-driven and knowledge-driven approaches is a logical next-step in the process. By making use of artificial intelligence (AI) techniques, edge and corner cases could be identified automatically. These cases are expected to help identify failures or strange behaviors of AVs, and consequently increase the assurance of safety of these vehicles. SDLS should therefore facilitate this. In this sense, M-SDL is currently the leading language in this direction.

Different scenarios are typically treated as being independent of each other and are not characterized by their relative level of complexity. Such characterization will become more important for test engineers and regulators to identify edge and corner cases properly. At the same time, an agreed-upon measure of complexity will aid scenario designers to classify the difficulty of their scenarios in the same way the automation level of a vehicle is classified into one of six levels [50]. The level of difficulty could be included in each scenario in the SDLS.

Many different countries will be developing their own AV regulations and policies, so a common understanding of the terminologies used in different languages is critical to the car manufacturers. Currently, *stiEF* is the only multilingual SDL that is designed to overcome this problem. More SDLS that takes the issues involved in multilingual documents will be beneficial.

## 5. Conclusions

In the past decade, the direction for testing and validation of the functionalities of AVs shifted from a distance-based to a scenario-based approach. A number of SDLS were developed to support this. In this paper, we reviewed and compared their features. Since they are designed to fulfill different goals, their characteristics are different. Their design philosophy and approaches are also different. While some make use of a hierarchical structure to describe a scene and the actors, others use other de facto standards such as openDRIVE for road description. The full scenario is composed of all such descriptions. In terms of data format, XML is the most commonly used among the SDLS reviewed. Three of them, Hisperia, SCENIC and M-DSL, make use of a programming language format that potentially enable them to describe dynamic scenes more easily.

An important trend is that these software are open sourced (except Hisperia). This is important, as it allows these tools to be easily acquired and utilized by industry and academia. Open source also allows users to modify and tailor the software to their needs. In this way, the longevity of these tools are ensured.

A scenario that is precisely specified will be of little use by itself. It is important to note which simulators are able to import scenarios from which SDLS. At this time, most simulation software can only import scenarios described by one or two SDLS. The exception is CARLA which is itself an open-source simulation platform. Ultimately, the choice of SDL may be determined by the choice of simulator, since most simulators are not open-sourced.

## References

1. Hancock, P.A.; Nourbakhsh, I.; Stewart, J. On the future of transportation in an era of automated and autonomous vehicles. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 7684–7691. [CrossRef]
2. Hawkins, A. Waymo's Driverless Car: Ghost-Riding in the Back Seat of a Robot Taxi. 2019. Available online: https://www.theverge.com/2019/12/9/21000085/waymo-fully-driverless-car-self-driving-ride-hail-service-phoenix-arizona (accessed on 1 April 2021).
3. Piper, K. It's 2020. Where Are Our Self-Driving Cars? 2020. Available online: https://www.vox.com/future-perfect/2020/2/14/21063487/self-driving-cars-autonomous-vehicles-waymo-cruise-uber (accessed on 1 April 2021).
4. Kalra, N.; Paddock, S.M. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability. *Transp. Res. Part A* **2016**, *94*, 182–193.
5. Menzel, T.; Bagschik, G.; Maurer, M. Scenarios for development, test and validation of automated vehicles. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1821–1827.

6.    Menzel, T.; Bagschik, G.; Isensee, L.; Schomburg, A.; Maurer, M. From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 2383–2390.

7.    Neurohr, C.; Westhofen, L.; Henning, T.; de Graaff, T.; Möhlmann, E.; Böde, E. Fundamental Considerations around Scenario-Based Testing for Automated Driving. *arXiv* **2020**, arXiv:2005.04045.

8.    ISO-26262. *Road Vehicles–Functional Safety*; Standard, International Organization for Standardization: Geneva, Switzerland, 2016.

9.    Andreotti, E.; Boyraz, P.; Selpi, S. Mathematical Definitions of Scene and Scenario for Analysis of Automated Driving Systems in Mixed-Traffic Simulations. *IEEE Trans. Intell. Veh.* **2021**, *6*, 366–375. [CrossRef]

10.   Go, K.; Carroll, J.M. The blind men and the elephant: Views of scenario-based system design. *Interactions* **2004**, *11*, 44–53. [CrossRef]

11.   Geyer, S.; Baltzer, M.; Franz, B.; Hakuli, S.; Kauer, M.; Kienle, M.; Meier, S.; Weißgerber, T.; Bengler, K.; Bruder, R.; 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.* **2013**, *8*, 183–189. [CrossRef]

12.   Ulbrich, S.; Menzel, T.; Reschka, A.; Schuldt, F.; Maurer, M. Defining and substantiating the terms scene, situation, and scenario for automated driving. In Proceedings of the 18th IEEE International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 15–18 September 2015; pp. 982–988.

13.   Elrofai, H.; Paardekooper, J.P.; de Gelder, E.; Kalisvaart, S.; den Camp, O.O. Scenario-based safety validation of connected and automated driving. In *Technical Report*; Netherlands Organization for Applied Scientific Research: Hague, The Newtherlands, 2018.

14.   De Gelder, E.; Paardekooper, J.P.; Saberi, A.K.; Elrofai, H.; Ploeg, J.; Friedmann, L.; De Schutter, B. Ontology for scenarios for the assessment of automated vehicles. *arXiv* **2020**, arXiv:2001.11507.

15.   Zhang, X.; Khastgir, S.; Jennings, P. Scenario Description Language for Automated Driving Systems: A Two Level Abstraction Approach. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 973–980.

16.   Bagschik, G.; Menzel, T.; Maurer, M. Ontology based scene creation for the development of automated vehicles. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1813–1820.

17.   Reschka, A.; Böhmer, J.R.; Gacnik, J.; Köster, F.; Wille, J.M.; Maurer, M. Development of software for open autonomous automotive systems in the Stadtpilot-project. In Proceedings of the WIT 2011 8th International Workshop on Intelligent Transportation, Hamburg, Germany, 22–23 March 2016.

18.   Bock, J.; Krajewski, R.; Eckstein, L.; Klimke, J.; Sauerbier, J.; Zlocki, A. Data basis for scenario-based validation of HAD on highways. In Proceedings of the 27th Aachen Colloquium Automobile and Engine Technology, Aachen, Germany, 8–10 October 2018.

19.   Körner, S.J. *RECAA-Werkzeugunterstützung in der Anforderungserhebung*; KIT Scientific Publishing: Karlsruhe, Germany, 2014.

20.   Bock, F.; Sippl, C.; Heinzz, A.; Lauerz, C.; German, R. Advantageous usage of textual domain-specific languages for scenario-driven development of automated driving functions. In Proceedings of the 2019 IEEE International Systems Conference (SysCon), Orlando, FL, USA, 8–11 April 2019; pp. 1–8.

21.   Franke, K. Volkswagen Group: Leveraging VIRES VTD to Design a Cooperative Driver Assistance System. In *Engineering Reality Magazine, Winter 2018*; Volkswagen AG: Wolfsburg, Germany, 2018; Volume VIII, pp. 10–14.

22.   ASAM. OpenSCENARIO. 2021. Available online: https://www.asam.net/standards/detail/openscenario/ (accessed on 1 December 2021).

23.   Hause, M. The SysML modeling language. In Proceedings of the 15th European Systems Engineering Conference, Edinburgh, UK, 18–20 September 2006; Volume 9, pp. 1–12.

24.   ASAM. ASAM OpenCRG. 2021. Available online: https://www.asam.net/standards/detail/opencrg/ (accessed on 1 December 2021).

25.   Natick, Massachusetts. MATLAB Version(R2021a). 2021. Available online: https://www.mathworks.com/ (accessed on 1 December 2021).

26.   CARLA Team. CARLA Simulator. 2021. Available online: https://carla.org (accessed on 1 December 2021).

27.   Siemens. Simcenter Prescan. 2021. Available online: https://tass.plm.automation.siemens.com/prescan-overview (accessed on 1 December 2021).

28.   Berger, C. *Automating Acceptance Tests for Sensor-and Actuator-Based Systems on the Example of Autonomous Vehicles*; Technical Report AIB-2010-16; Department of Computer Science, RWTH Aachen University: Shaker, Germany, 2010.

29.   Berger, C.; Rumpe, B. Engineering autonomous driving software. *arXiv* **2014** arXiv:1409.6579.

30.   Schütt, B.; Braun, T.; Otten, S.; Sax, E. SceML: A graphical modeling framework for scenario-based testing of autonomous vehicles. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Virtual, 16–23 October 2020; pp. 114–120.

31.   OAS. Open Autonomous Safety. 2021. Available online: https://oas.voyage.auto/scenarios/ (accessed on 1 April 2021).

32.   Queiroz, R.; Berger, T.; Czarnecki, K. GeoScenario: An open DSL for autonomous driving scenario representation. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 287–294.

33.   OSM. Open Street Map (OSM). 2021. Available online: https://www.openstreetmap.org/ (accessed on 1 December 2021).

34.   Haklay, M.; Weber, P. Openstreetmap: User-generated street maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [CrossRef]

35. Bender, P.; Ziegler, J.; Stiller, C. Lanelets: Efficient map representation for autonomous driving. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium, Dearborn, MI, USA, 8–11 June 2014; pp. 420–425.
36. OSM. Java Open Street Map Editor. 2021. Available online: https://josm.openstreetmap.de/ (accessed on 1 December 2021).
37. Althoff, M.; Koschi, M.; Manzinger, S. CommonRoad: Composable benchmarks for motion planning on roads. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 719–726.
38. Klischat, M.; Dragoi, O.; Eissa, M.; Althoff, M. Coupling sumo with a motion planning framework for automated vehicles. In Proceedings of the SUMO User Conference, Berlin/Adlershof, Germany, 13–15 May 2019.
39. Poggenhans, F.; Pauls, J.H.; Janosovits, J.; Orf, S.; Naumann, M.; Kuhnt, F.; Mayr, M. Lanelet2: A high-definition map framework for the future of automated driving. In Proceedings of the 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 1672–1679.
40. Zheng, L.; Li, B.; Yang, B.; Song, H.; Lu, Z. Lane-level road network generation techniques for lane-level maps of autonomous vehicles: A survey. *Sustainability* **2019**, *11*, 4511. [CrossRef]
41. Fremont, D.J.; Dreossi, T.; Ghosh, S.; Yue, X.; Sangiovanni-Vincentelli, A.L.; Seshia, S.A. Scenic: A language for scenario specification and scene generation. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, Phoenix, AZ, USA, 22–26 June 2019; pp. 63–78.
42. Gordon, A.D.; Henzinger, T.A.; Nori, A.V.; Rajamani, S.K. Probabilistic programming. In *Future of Software Engineering Proceedings*; Association for Computing Machinery: New York, NY, USA, 2014; pp. 167–181. [CrossRef]
43. Goodman, N.; Mansinghka, V.; Roy, D.M.; Bonawitz, K.; Tenenbaum, J.B. Church: A language for generative models. *arXiv* **2012**, arXiv:1206.3255.
44. Dreossi, T.; Fremont, D.J.; Ghosh, S.; Kim, E.; Ravanbakhsh, H.; Vazquez-Chanlatte, M.; Seshia, S.A. Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2019; Volume 11561. [CrossRef]
45. Michel, O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, 5. [CrossRef]
46. Advanced Platform Team, LG Electronics America R&D Lab. LGSVL Simulator. 2021. Available online: https://www.lgsvlsimulator.com (accessed on 1 December 2021).
47. Foretellix, Inc. Measurable Scenario Description Language. 2021. Available online: https://www.foretellix.com/open-language/ (accessed on 1 December 2021).
48. Foretellix, Inc. M-SDL Seminar. 2020. Available online: https://www.youtube.com/watch?v=puM8v8KIK2k (accessed on 1 December 2021).
49. Baidu Cloud. Apollo: Autonomous Driving Solution. 2021. Available online: https://apollo.auto (accessed on 1 December 2021).
50. U.S. Department of Transportation. SAE International's Levels of Automation for Defining Driving Automation in On-Road Motor Vehicles. 2016. Available online: https://www.sae.org/news/3544/ (accessed on 1 December 2021).