ASAM OpenX

Standardization for validation and verification of autonomous driving

Benjamin Engel
ASAM e.V.

Nicco Hagedorn
ASAM e.V.

8. Oktober 2021

ASAM
Association for Standardization of
Automation and Measuring Systems

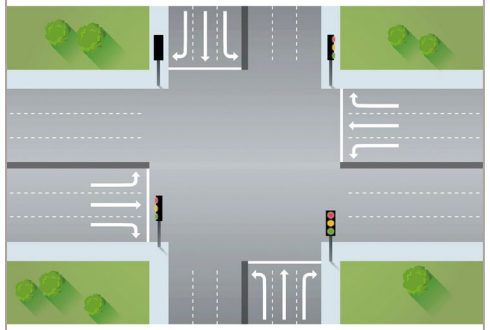
ASAM OpenX

Overview and Status Update

ASAM

An Overview of ASAM OpenX…

ASAM OpenDRIVE®

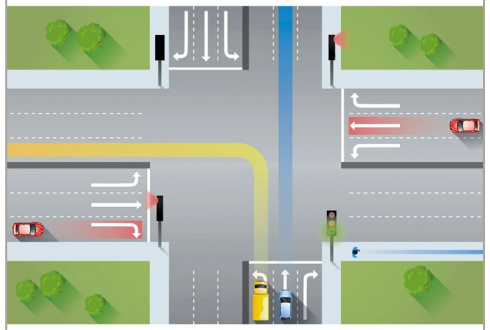


Static Road Network Desription

ASAM OpenCRG®

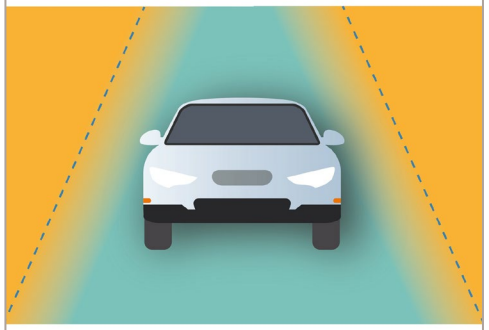
Static Road Surface Desription

ASAM OpenSCENARIO®



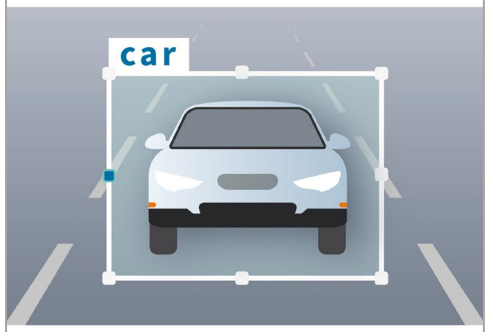
Dynamic Scenario Desription

ASAM OpenODD



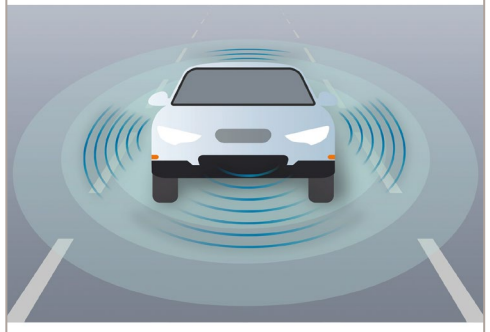
Defining the Operational Design Domain for Automated Vehicles

ASAM OpenLABEL



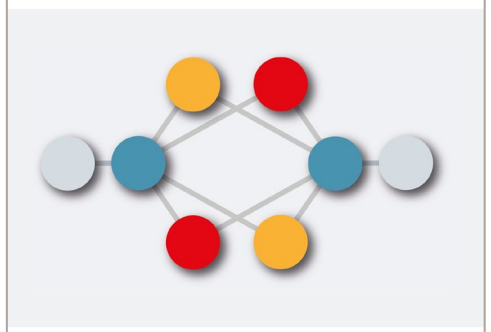
Standardized Labeling for objects and Scenarios

ASAM OSI®



Interface for Simulation

ASAM OpenXOntology



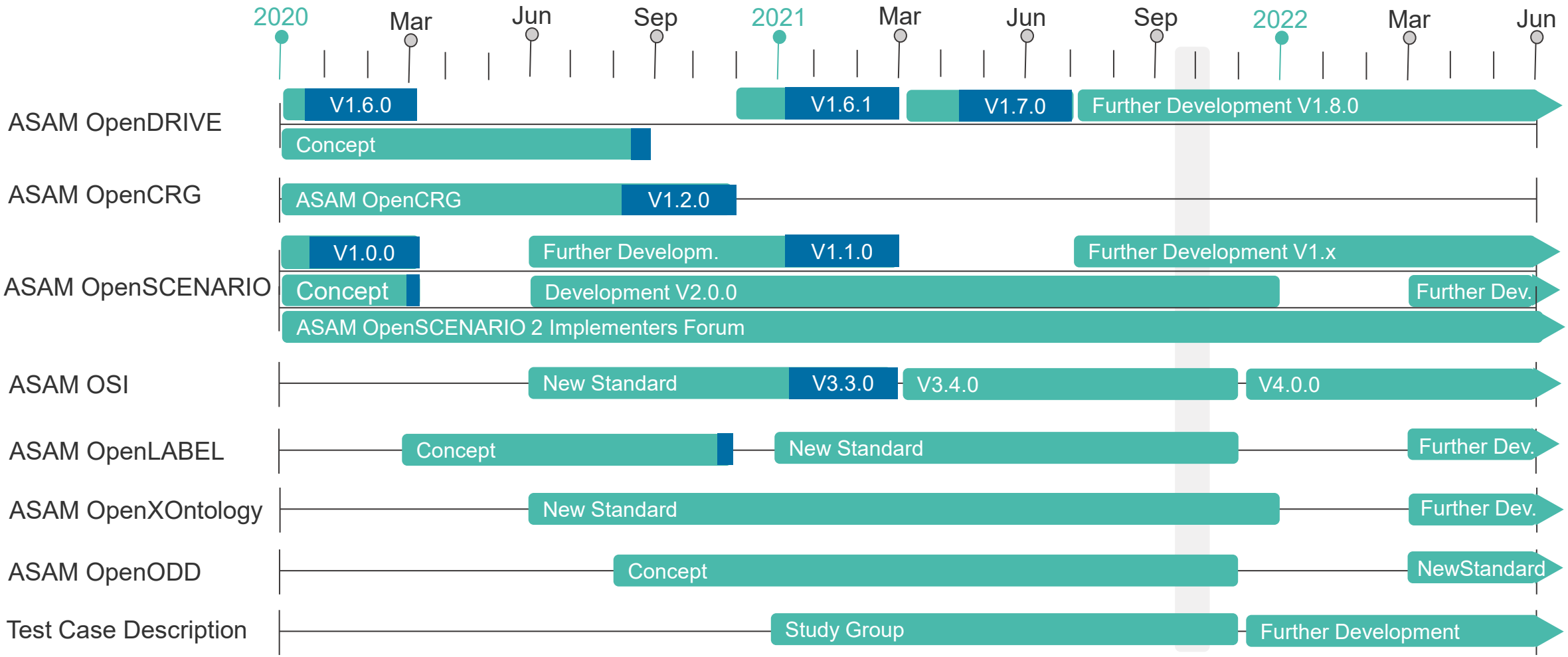
Core Domain Model for the ASAM Simulation Domain

Test Specification Study Group

Analysis and harmonization of automotive testing techniques and standards

ASAM

Project Plan



ASAM OpenDRIVE

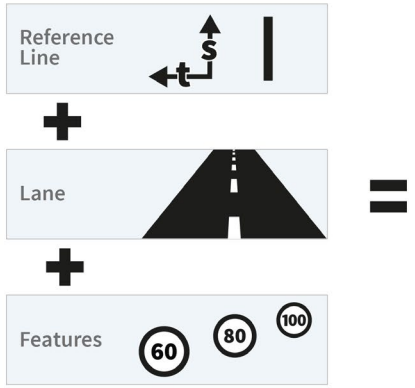
- File format for **the description of road networks.**

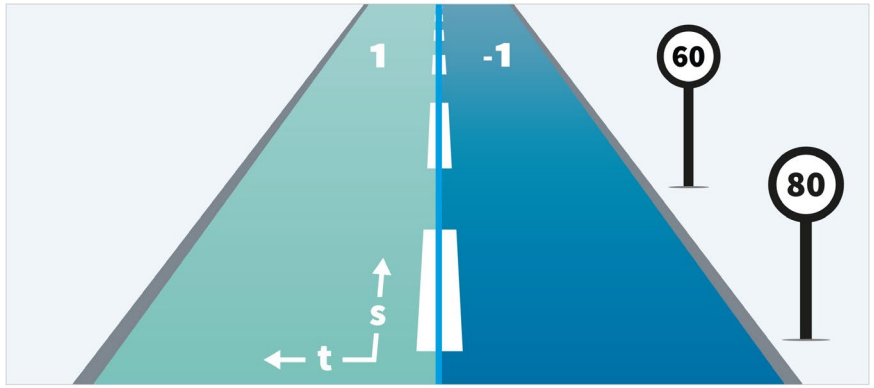
- V1.6.1 released March 2021
  - Integration of OpenCRG
  - Bug fixes & clarifications

  - New project started in May 2021 [Proposal]
    - Two releases:

    - V1.7.0 in July 2021
      - Address known issues in V1.6.1

    - V1.8.0 in November 2022
      - New features based on concepts from the OpenDRIVE Concept Paper

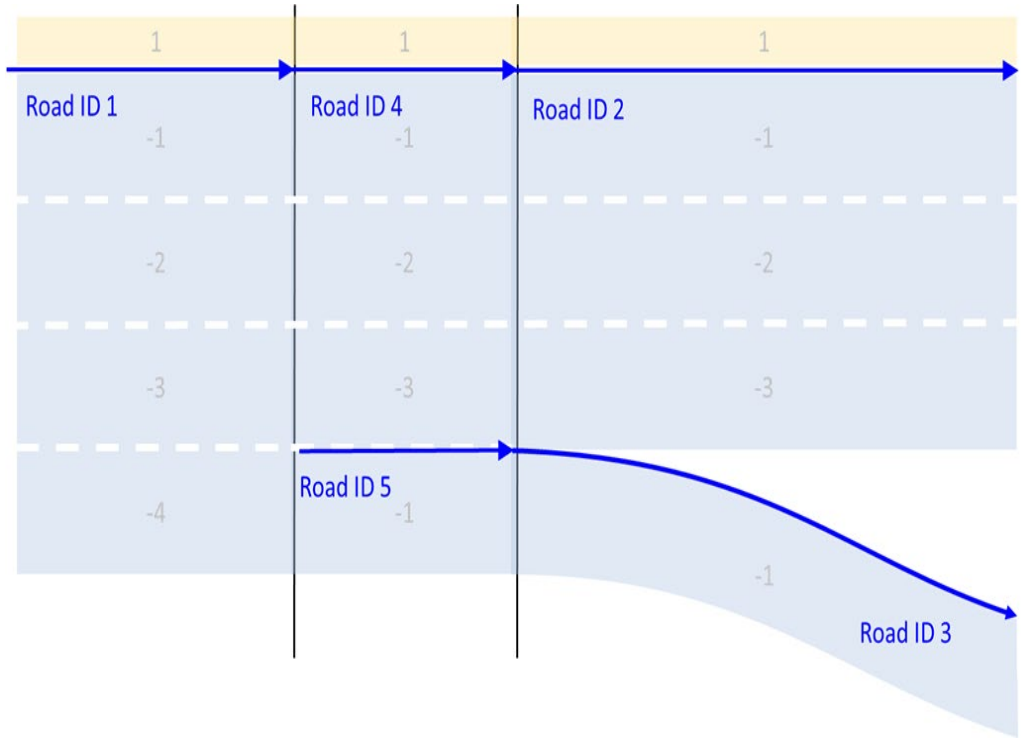




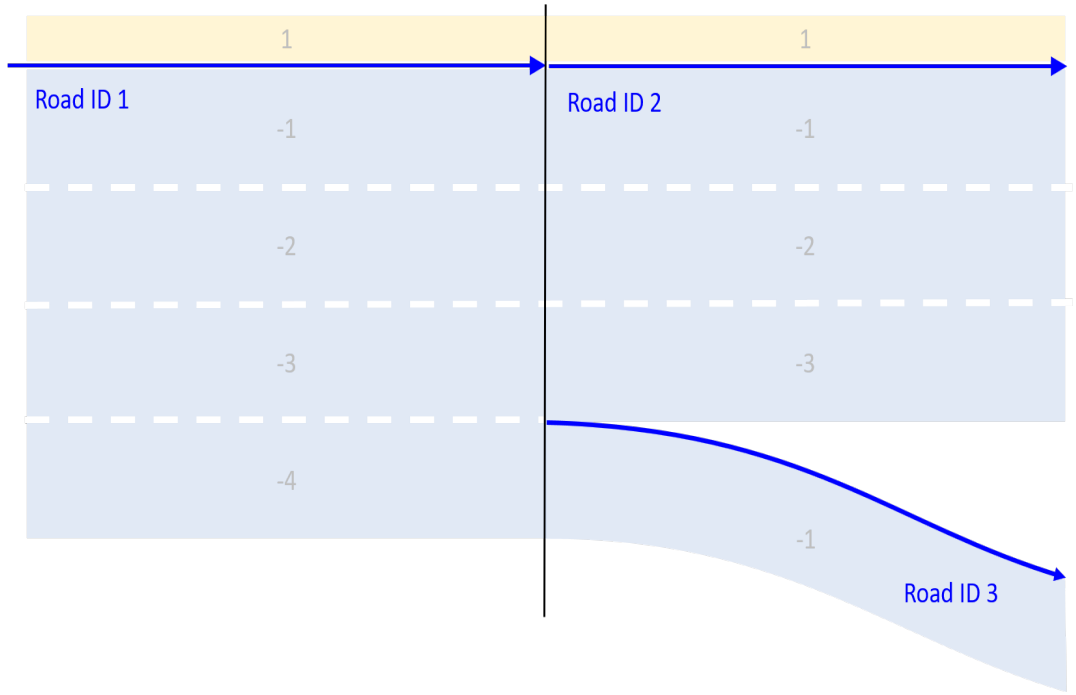
What's new - ASAM OpenDRIVE 1.7

Direct Junctions

used for a typical motorway exit.



ASAM OpenDRIVE 1.6

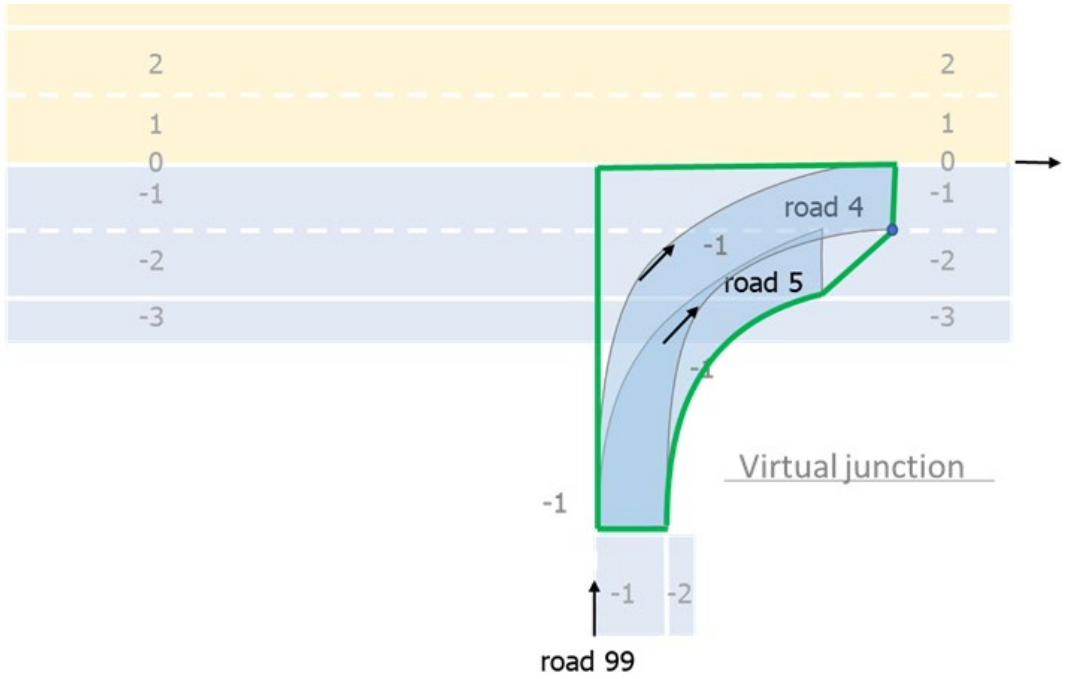


ASAM OpenDRIVE 1.7

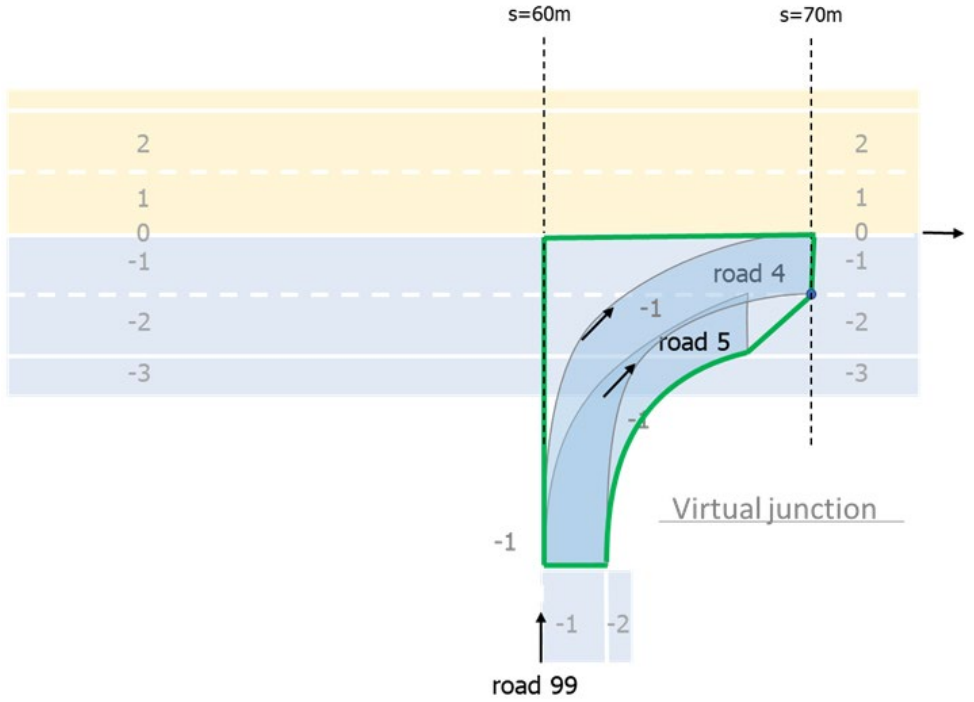
What's new - ASAM OpenDRIVE 1.7

Virtual Junctions

used for driveways and supermarket entries



ASAM OpenDRIVE 1.6



ASAM OpenDRIVE 1.7

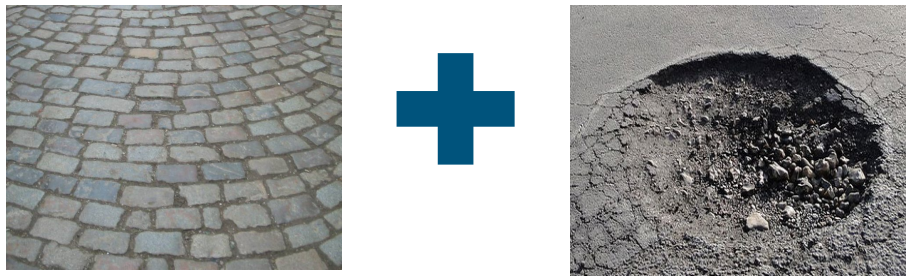
What's new - ASAM OpenDRIVE 1.7

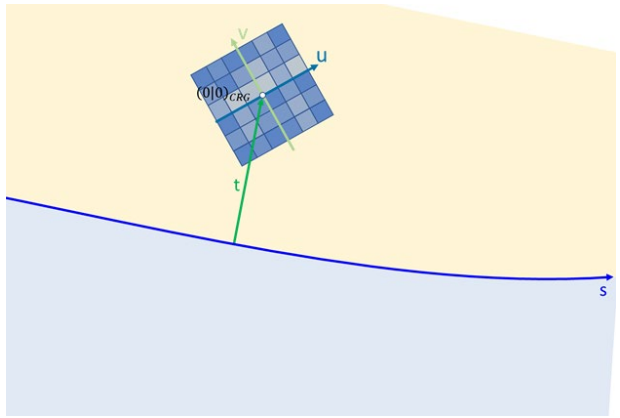
OpenCRG for objects

Required for potholes, manholes, cracks patches

In OpenDRIVE 1.7

- Can have road surface and a pothole at the same place
- It is clear with ASAM OpenCRG belongs to which object
- Object with OpenCRG can be rotated





Detailed plans for ASAM OpenDRIVE 1.8

OpenDRIVE 1.8.0 (Release November 2022)

UML Model Improvement 1.8.0
Want to find a technical solution on validating certain checks via the schema files.
Goal to embed more rules into the model

Junction Modelling
Junction modelling: junction road and junction area.
Junction guideline – There are several ways to model an existing junction to OD junctions.

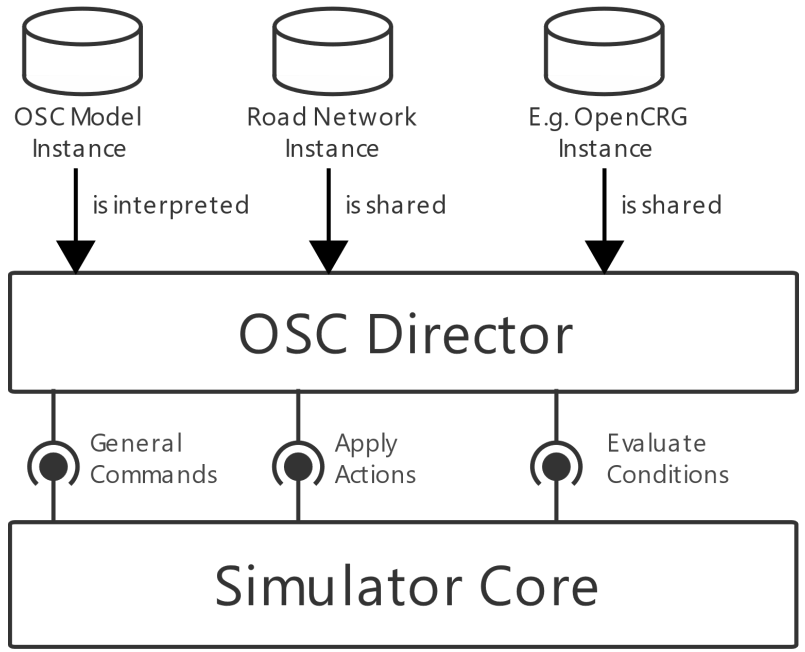
Road Geomentry
A better spline to avoid small lines.
Crossfalls for town roads.
superelevation and roadShape ?

Environmental Modelling
External Object Reference
Interface to other GIS Standards
Object Lists – and compatible to OSI and OpenLABEL

ASAM OpenSCENARIO V1.x

- File format for the **description of dynamic content in driving simulation applications**.

- V1.1.0 released March 2021
  - Many clarifications in the specification
  - Complete restructure of the documentation
  - Support for logical scenarios (Coverage)
  - More flexible maneuver modelling
  - 100% backward compatible to OpenSCENARIO v1.0.0



Improved runtime model of OSC1.x

ASAM OpenSCENARIO V1.x

- Project for V1.X started on 03.09!
  → Proposal

- Topics being addressed:

1. Actions
   - A more abstract way to define actions independent of e.g. the road network.
   - More actions to OpenSCENARIO for parking, weather conditions, traffic and non-movement actions (e.g. open door of a vehicle).

2. Parameters
   - Scenario distributions across parameter ranges

3. Runtime
   - Further detail the system boundaries, i.e. what is under the responsibility of different simulation components and how they relate to the OpenSCENARIO standard

ASAM OpenSCENARIO 2 Implementers Forum

- **A platform for implementers and tool vendors to develop a shared understanding of the standard**

  1. Using the language to describe scenarios

  2. Implementing support for the language in tools

- Parallel to the development project

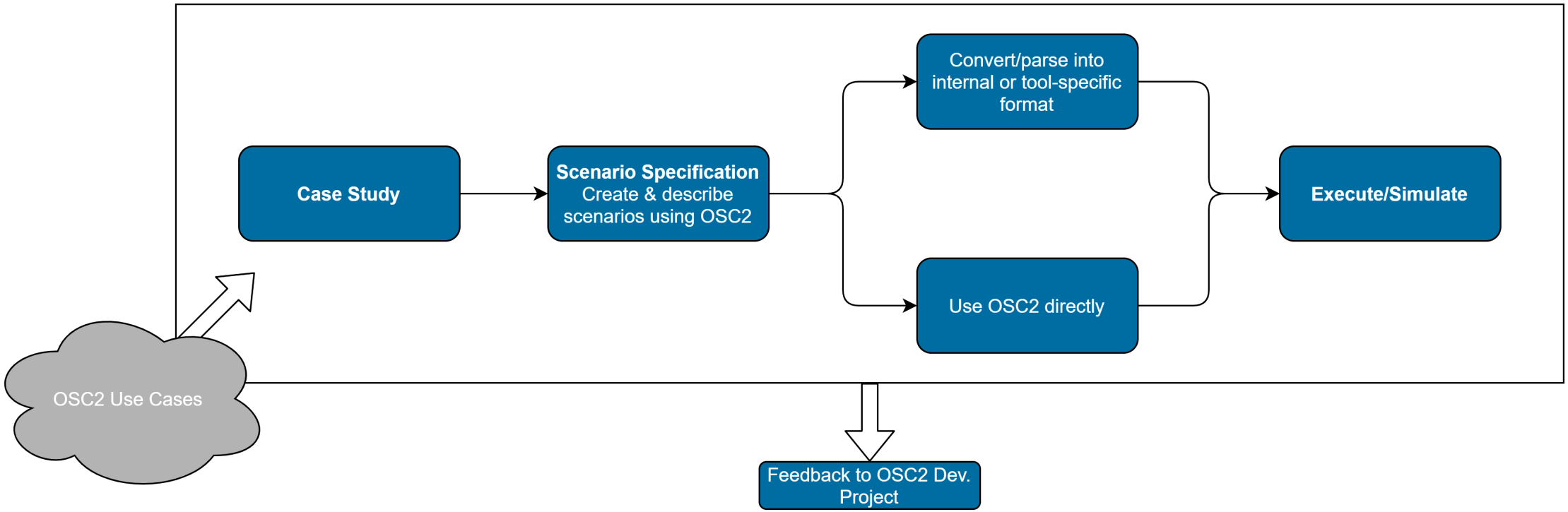
- Working mode is split according to different participant interests
  → Participants are able to provide input where they have expertise

- Feedback will regularly be exchanged with the development project

→ **Ensures a well understood, already partially supported standard on release**

ASAM OpenSCENARIO 2 Implementers Forum

The implementers forum workflow

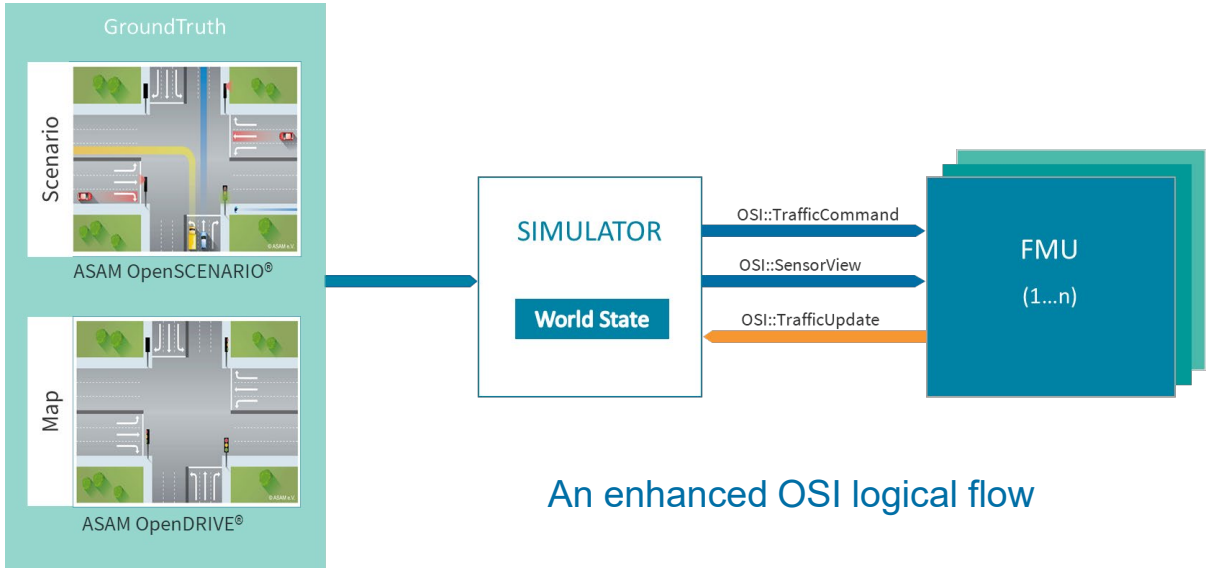


ASAM OSI (Open Simulation Interface)

- Open-source, generic **interface between models (e.g. sensors, traffic participants) and simulation environments**

- V3.3.0 released March 2021
  - Focus on defining an interface for Traffic Participants
  - TrafficCommand:: Event-based control of traffic participant models
  - TrafficUpdate:: Send updated properties of traffic participant models.

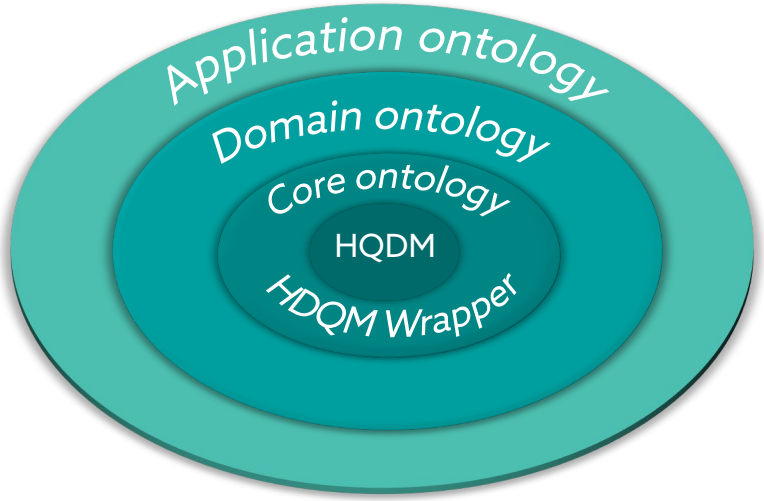
- V3.4.0 releasing Q1 2022
  - Focus on a complete rework of the documentation



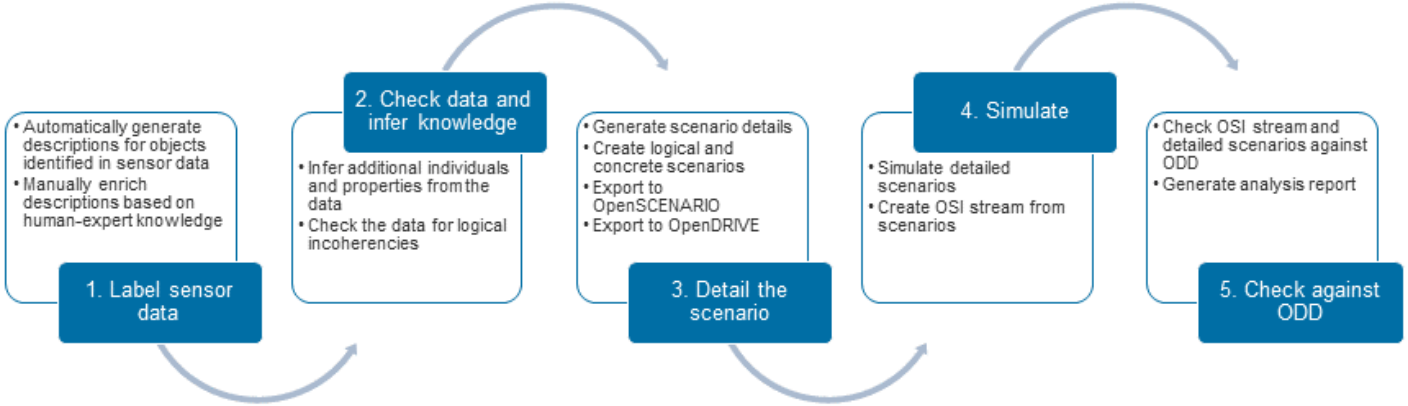
An enhanced OSI logical flow

ASAM OpenXOntology

- An ontology-based architecture for the domain concepts of on-road driving (road traffic, infrastructure, etc.) and thus a common definition of the domain model for the OpenX standards.
- V1.0.0 planned for November 2021
- Minimal Working Product (MWP) shared with OpenX project groups for review
  → How to use the Ontology together with OpenX standards in different workflows



The ontology architecture

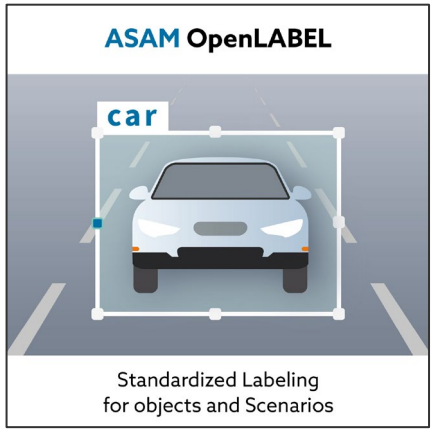


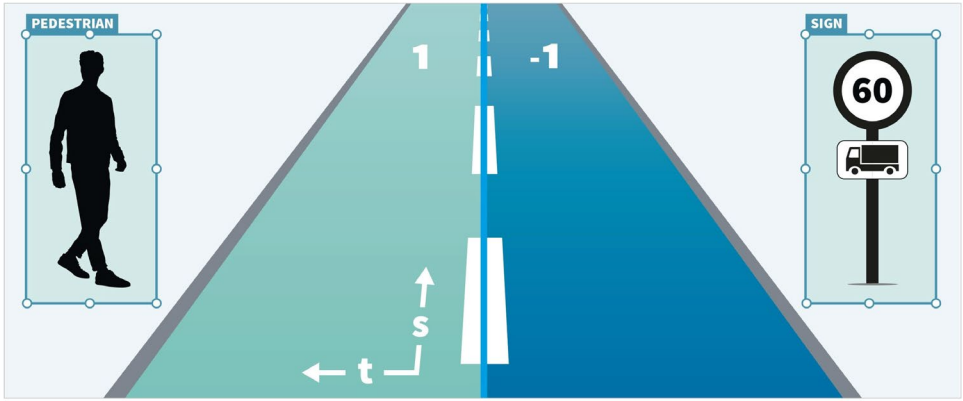
The workflow of the MWP (each step addresses a separate use case)

ASAM OpenLABEL

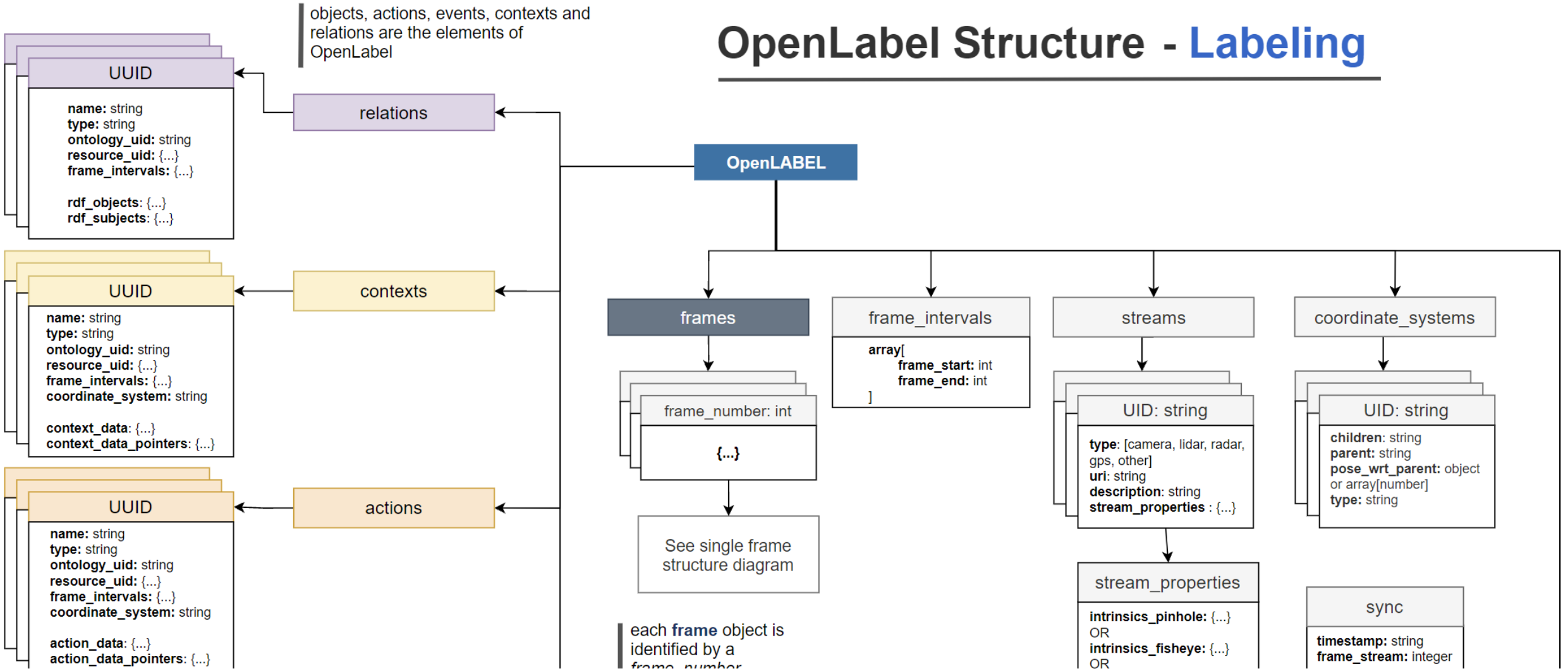
New Standard Initiative

- How to label → **Labeling formats for objects of interest and scenario data**
- Concept paper released November 2020 – [Link]
- Standard development project started Dec. 2020 – [Proposal Link]
  → V1.0.0 releasing November 2021

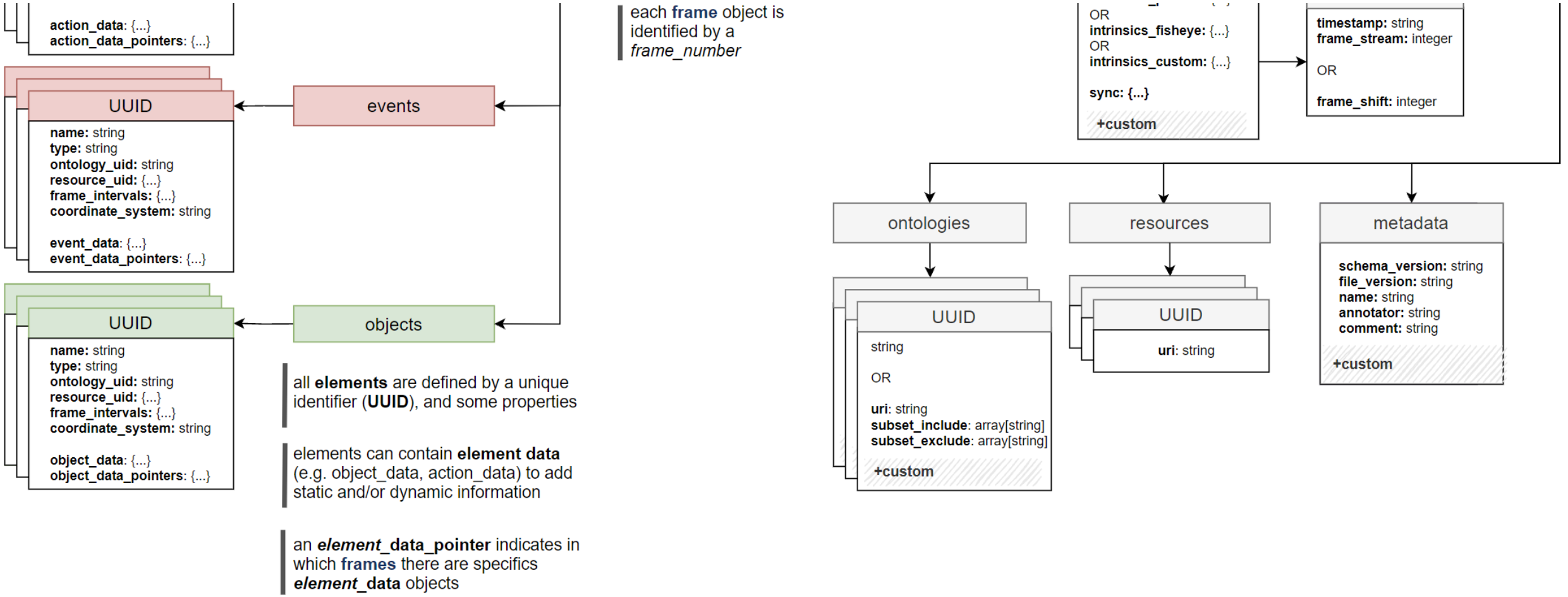




ASAM OpenLABEL Object Labeling

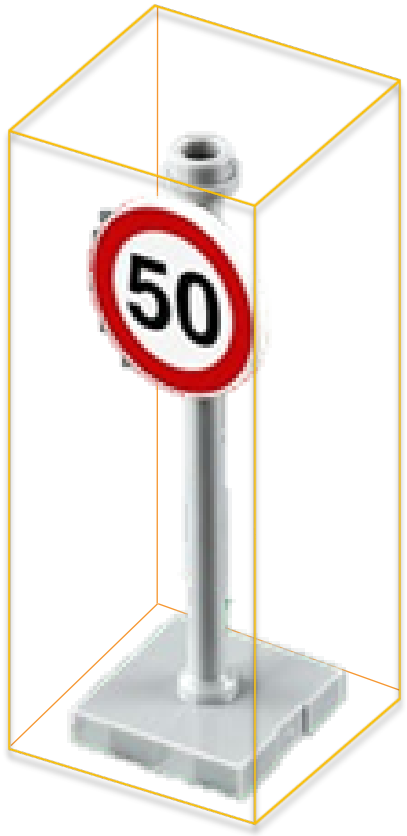


ASAM OpenLABEL Object Labeling

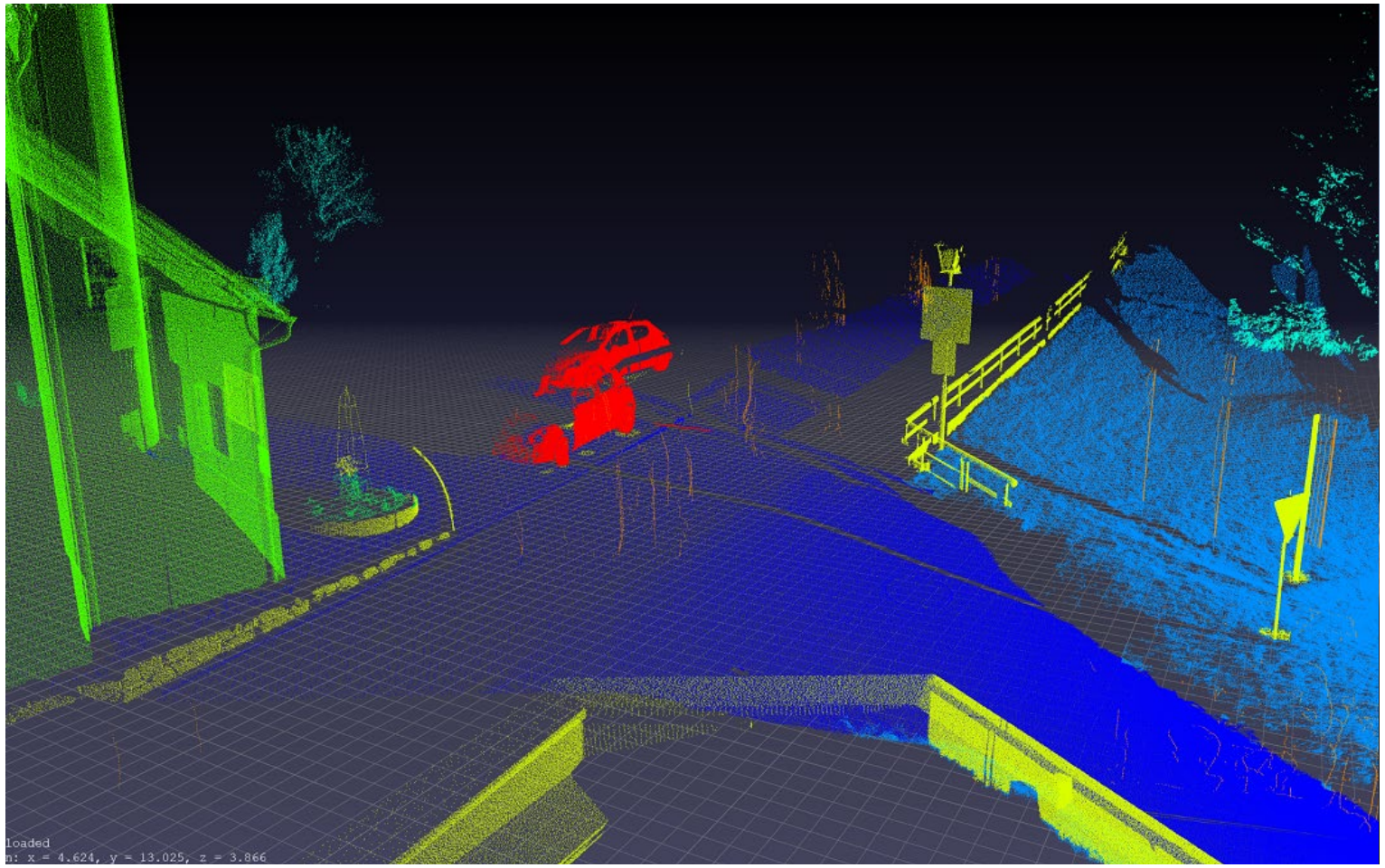


ASAM OpenLABEL: Object labeling

```
{ "openlabel": {
     "metadata": {
        "schema_version": "1.0.0" },
     "objects": {
        uuid: { "name": "sign_speedlimit_50",
                "type": "traffic_sign",
                "object_data": {
                   "cuboid": [{
                      "name": "sign_speedlimit_50",
                      "val": [x, y, z,
                              qa, qb, qc, qd,
                              sx, sy, sz],
                   "attributes":{
                      "boolean": [{
                         "name": "visible",
                         "val": true }],
                   }]
                }
           }
        }
     }
  }
```

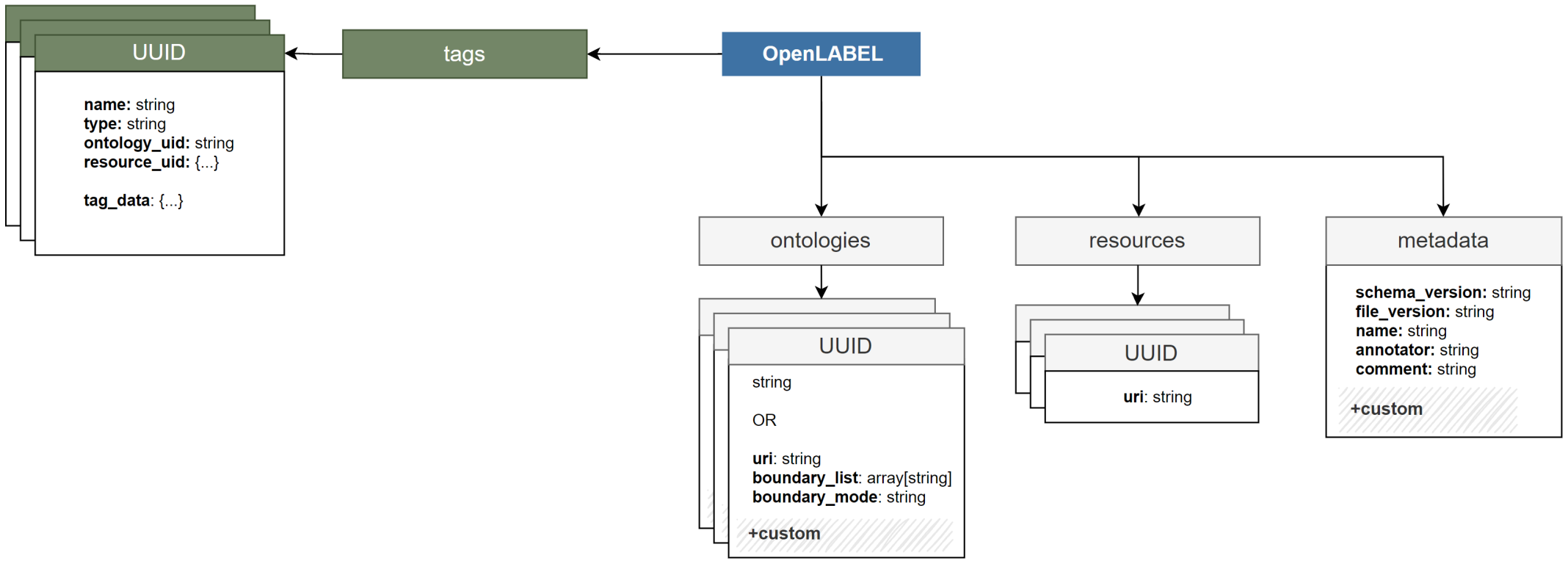


ASAM OpenLABEL Examples

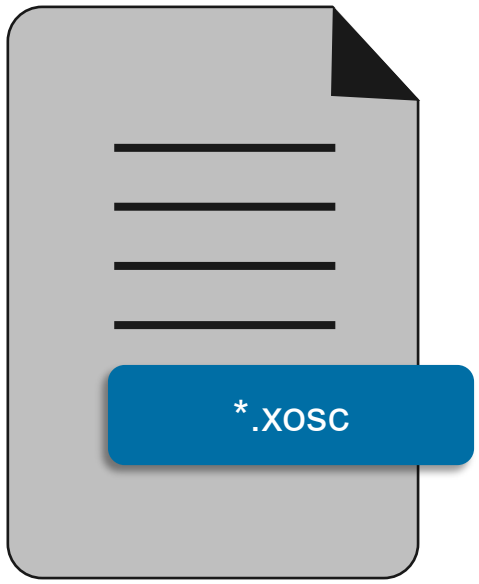


ASAM OpenLABEL Examples

OpenLabel Structure - Tagging

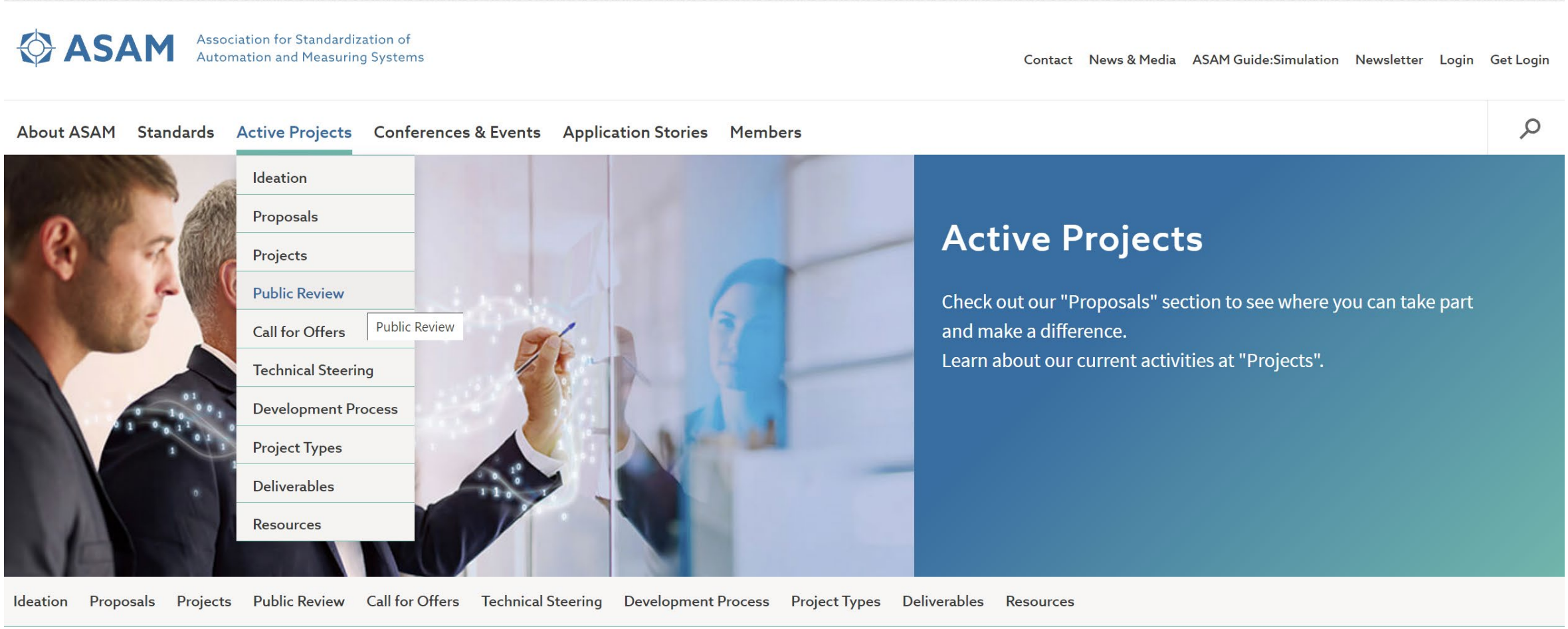


ASAM OpenLABEL: Scenario tagging



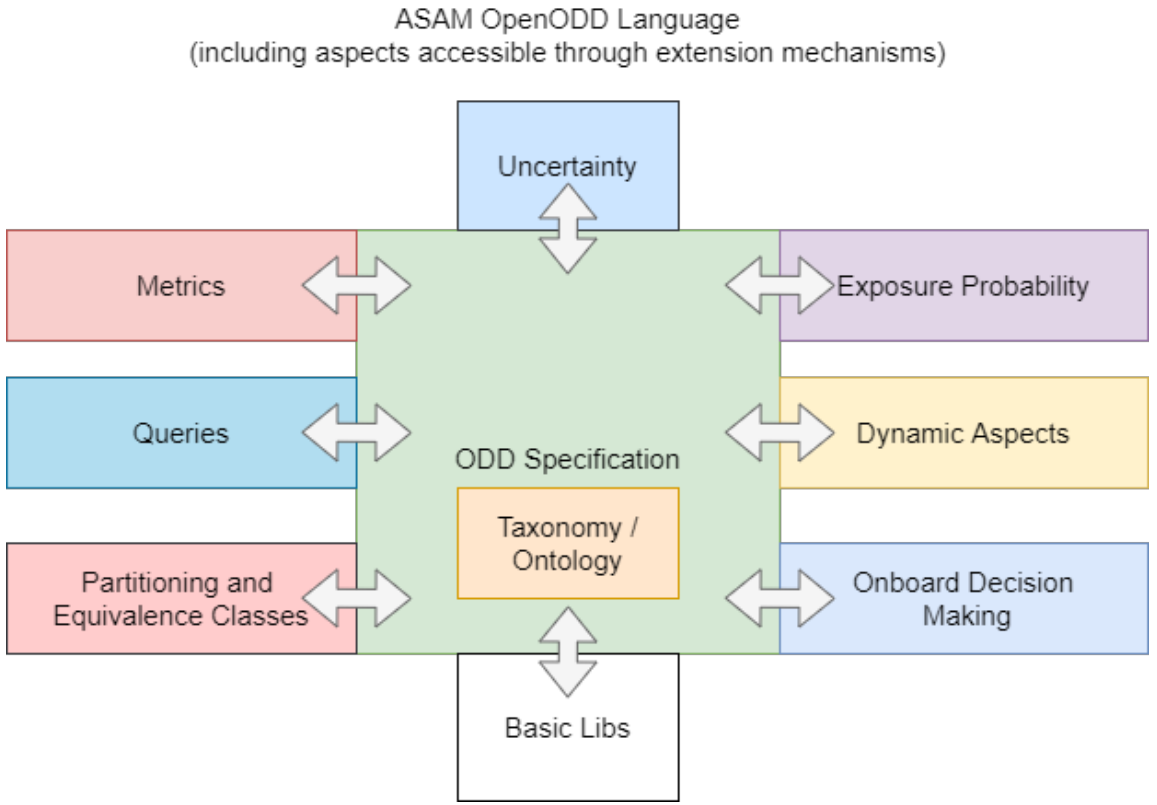

```
{
  "openlabel": {
    "metadata": {
      "schema_version": "1.0.0" },
    "ontologies": {
      "0": {
        "uri": "<link to an ontology>",
        "boundary_list": ["JunctionIntersection", "JunctionRoundabout"],
        "boundary_mode": "include"
      }
    },
    "tags": {
      "0": {
        "type": "JunctionIntersection",
        "ontology_uid": "0"
      }
    }
  }
}
```

ASAM OpenLABEL Review

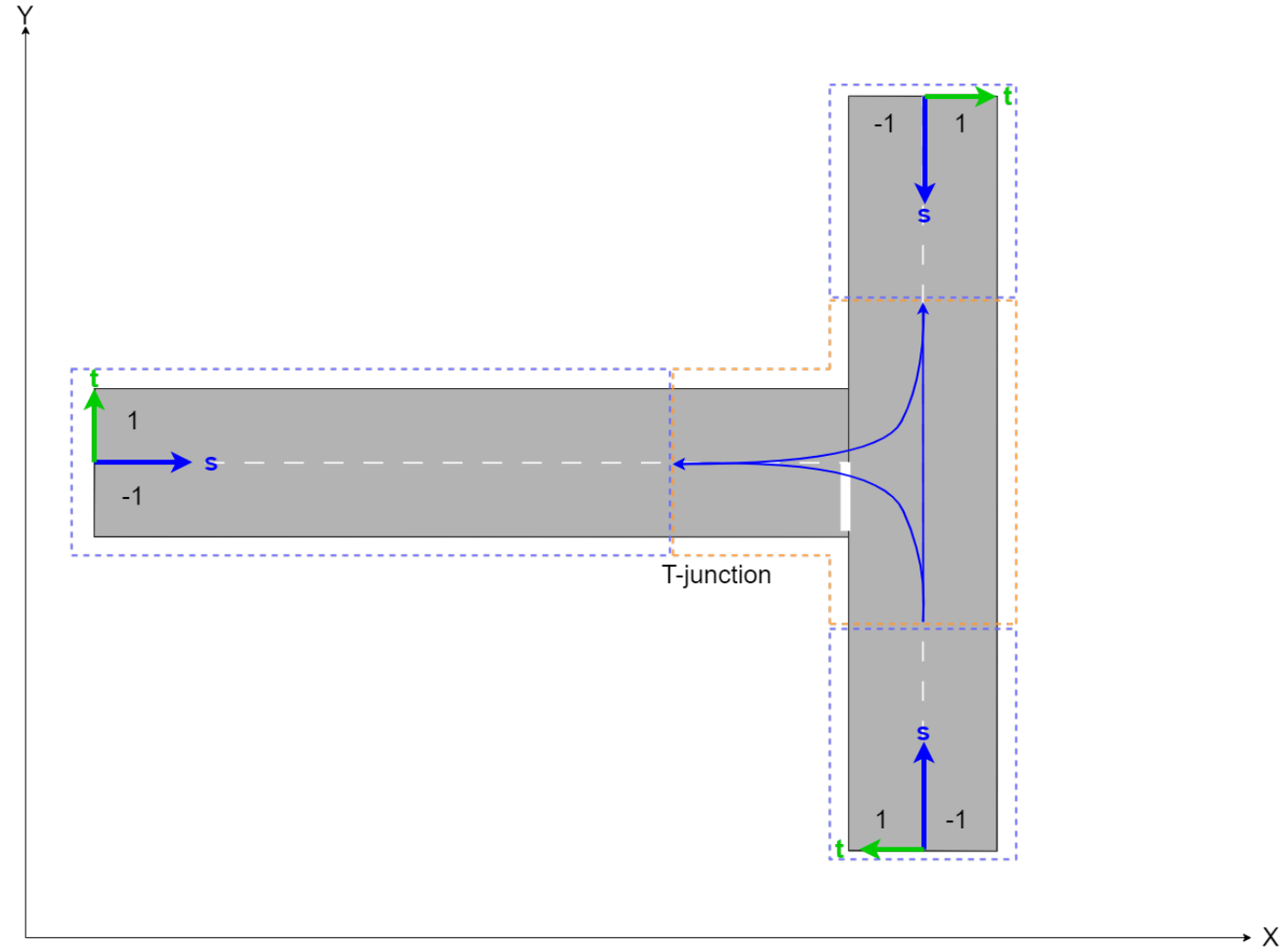


ASAM OpenODD

- Standardized format for the **definition of Operational Design Domains (ODDs)**
- Concept Project started on 9th September 2020 [Proposal link]
  → Concept paper releasing November 2021
- Aiming to start standard development project in March 2022

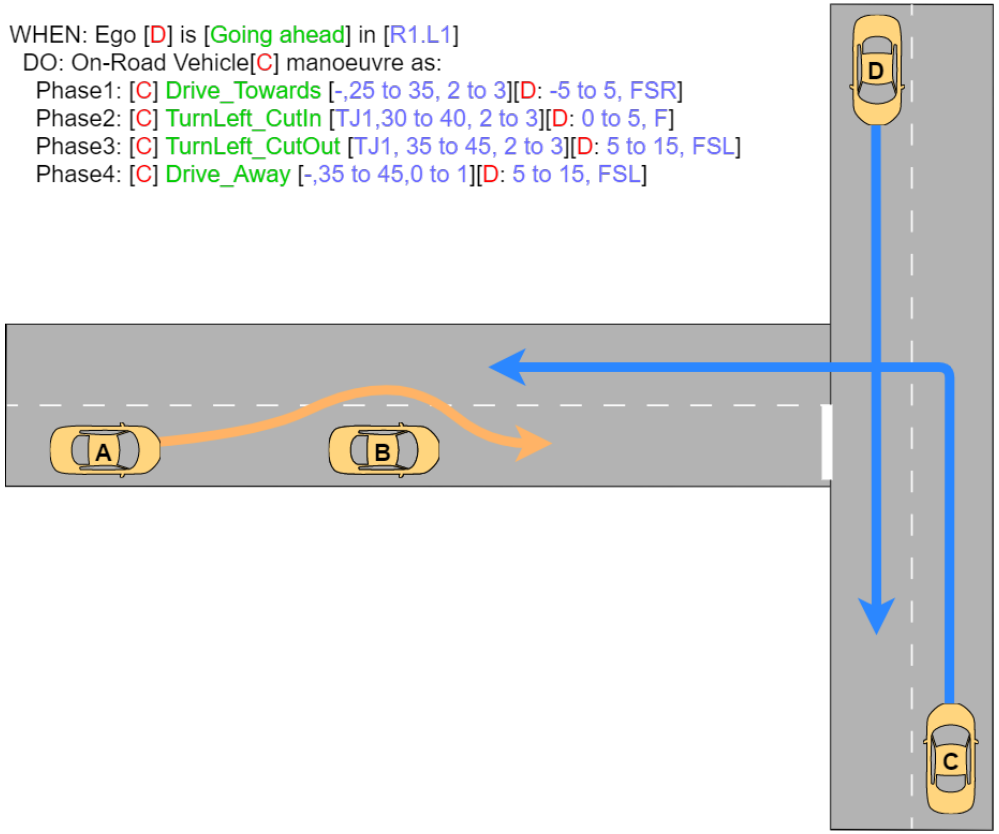


ASAM OpenODD Example

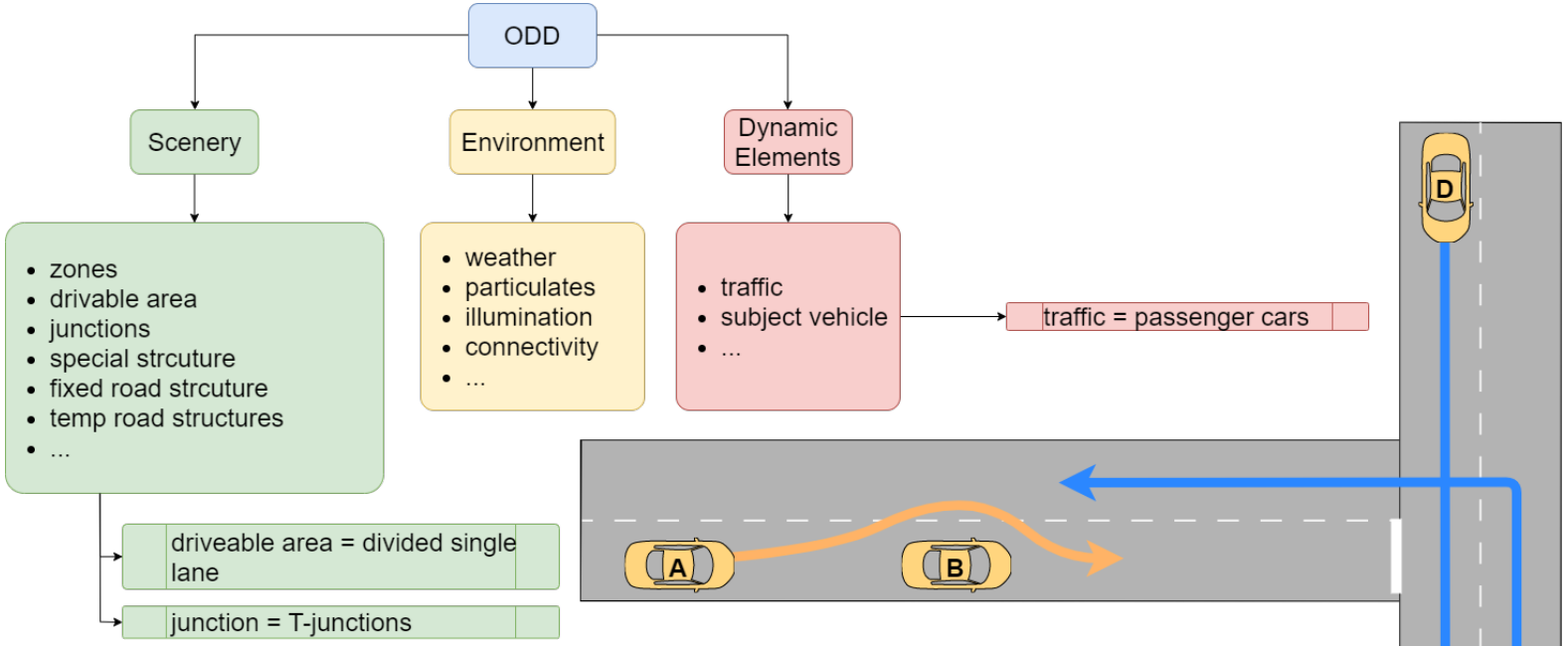


ASAM OpenODD Example

WHEN: Ego [D] is [Going ahead] in [R1.L1]
DO: On-Road Vehicle[C] manoeuvre as:
Phase1: [C] Drive_Towards [-,25 to 35, 2 to 3][D: -5 to 5, FSR]
Phase2: [C] TurnLeft_CutIn [TJ1,30 to 40, 2 to 3][D: 0 to 5, F]
Phase3: [C] TurnLeft_CutOut [TJ1, 35 to 45, 2 to 3][D: 5 to 15, FSL]
Phase4: [C] Drive_Away [-,35 to 45,0 to 1][D: 5 to 15, FSL]



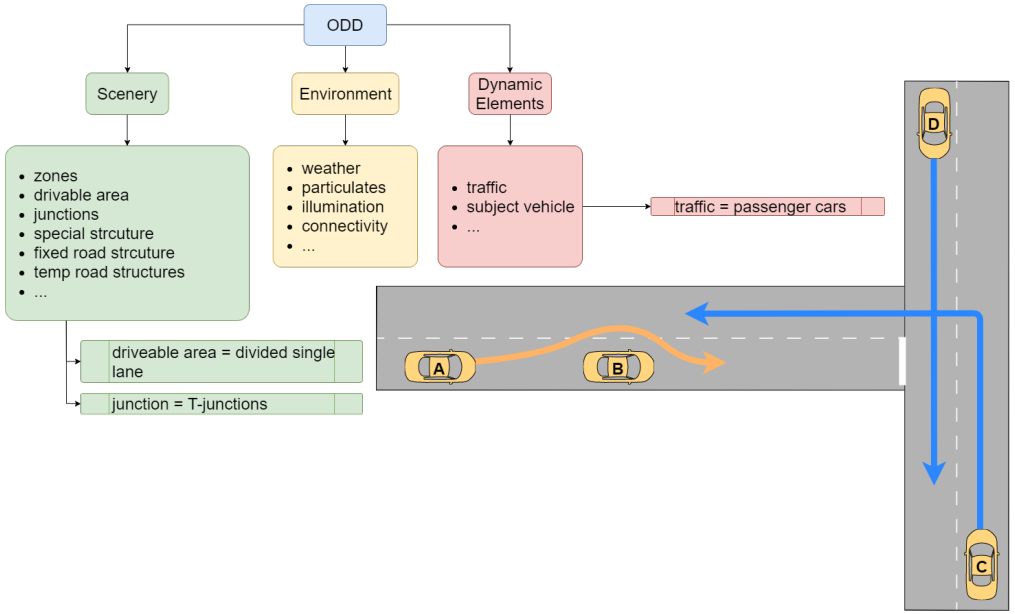
ASAM OpenODD Example



ASAM OpenODD Example

Example with pseudo code:

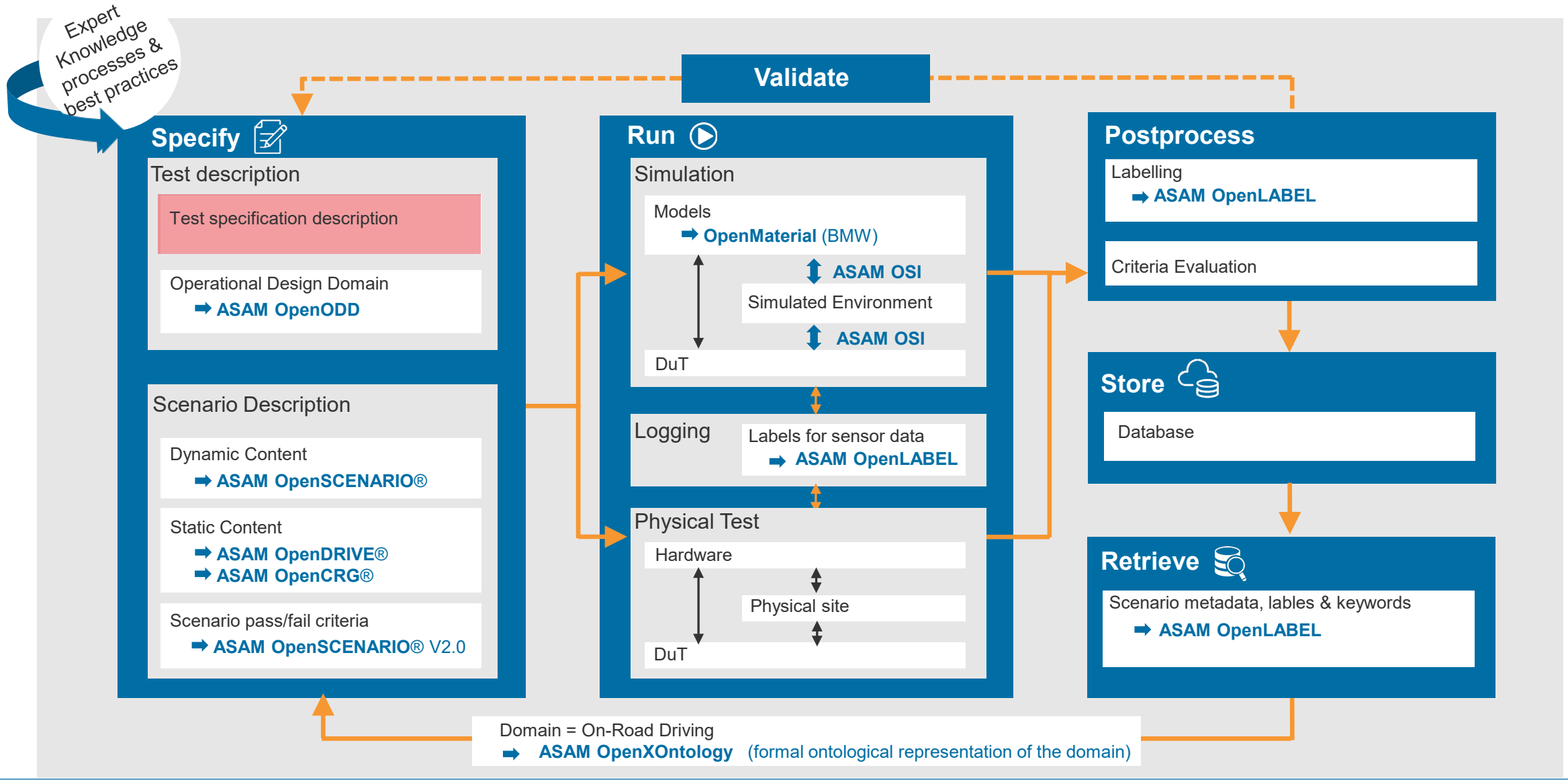
```
include odd_ontology

#restrictive definition
myODD(
  mode: ‘restrictive‘
  SUITABLE drivable_area WHEN ‘divided_single_lane‘;
  SUITABLE junction WHEN ‘T-junction‘;
  SUITABLE driving_direction WHEN ‘right_hand_traffic‘;
  SUITABLE laneDimensions WHEN {min: 3.0, max: 5.0};
  SUITABLE traffic WHEN ‘passenger cars‘;
  …
);
```



Scenario-Based Testing with OpenX

ASAM

Scenario-Based Testing With ASAM OpenX



ASAM

Test Specification Study Group

MISSION STATEMENT

- Examine the relevant test techniques and use cases for testing and homologation of the ADAS/AD Domain in automotive in detail.
  → Identify relevant standards, potential workflows and their variants, and the overall interplay between these parts to form a cohesive whole.

- Output: Documented set of overall use case for testing and homologation, a set of potential workflows implementing these, together with an overview of relevant users, standards and their application.

- Additionally, we will identify gaps in the workflows, leading to the identification of potentially needed additions to existing standards, liaisons between standards, or even the need for completely new standards.

- Recommendations for these standards or additions shall be collected and documented. The goal is to define a valid basis for follow up activities and projects

What is a Test Specification?

- ISO/IEC/IEEE 29119
  - complete documentation of the test design, test cases and test procedures for a specific test item

A Test Specification can be broken down into three components :

1. Test model
   - Describing the target of testing and the model used by the tester to derive test cases: traditionally „feature sets“ and „test coverage items“, more recently „test models“

2. Test case
   - Describing test cases – Specify preconditions, inputs, actions & expected results

3. Test procedure
   - Describing how test cases are grouped, plus additional information necessary to perform testing

Test Cases in Scenario-Based Testing

- Scenarios can be used as part of a testing approach to satisfy a test specification to varying degrees.
- So what is the relationship between a scenario and a test [case]?
  - Test input is scenario input road network, actors, trajectories
  - Test actions are scenario actions (what happens in a scenario)
  - Test results are a part of test results
- Which information goes into the test case? What goes into the scenario?
  - A scenario can include:
    - Behavioural definition
    - Sampling & calculation of complex parameters defined in the test
    - expected results
- Trade-off between mixing information and re-use of scenarios across tests
  → Ideally, aim for maximum reuse…

What is still to Come?

November 2021!

- Fully fledged report and analysis of the testing landscape
- Lots more detail on the various relationships between the components of tests
- Detailed definitions of relevant terms that take most perspectives into account
- Commentary by Technical Services on current and future regulation for testing and type approval
- User journeys → What methodologies are used to achieve which test goals? Testing Strategy Blueprint for e.g.:
  - Requirements-based HIL and SIL Testing
  - Scenario-based Testing on Proving Grounds
  - Test Data Management
  - Vehicle in the Loop
  - …
- Recommendations to the industry and standardization bodies – Where are there gaps in the workflows and how can these be harmonized?

ASAM OpenSCENARIO 2.0

WIP

Goals

- The remainder of this presentation will introduce **some** of the technical concepts in the upcoming ASAM OpenSCENARIO 2.0 standard

- It will **not** provide a thorough walkthrough of the standard and all technical details

- Soon after release we will be holding a series of deep-dive webinars to go through the standard and all content in depth

Disclaimer: The ASAM OpenSCENARIO 2.0 standard is not yet released. Content presented herein may change prior to release.

Introduction to ASAM OpenSCENARIO 2.0

- User-friendly, machine readable **Domain Specific Language (DSL) for scenario descriptions**

- **V2.0.0** releasing November 2021

- Deliverables:
  1. Reference documentation for the underlying domain model
  2. Guidelines on extending the domain model
  3. A reference manual documenting the language
  4. >20 tool independent workflows for use cases supported by the standard
  5. Guidelines on migrating from OpenSCENARIO 1.0
  6. Style guidelines for writing scenarios using OSC 2.0

- Current status:
  - Applying the finishing touches…

  - October: Review opens to ASAM membership

Introduction to ASAM OpenSCENARIO 2.0

A simplified view of abstraction levels as supported by OSC2… [Neurohr et al.]

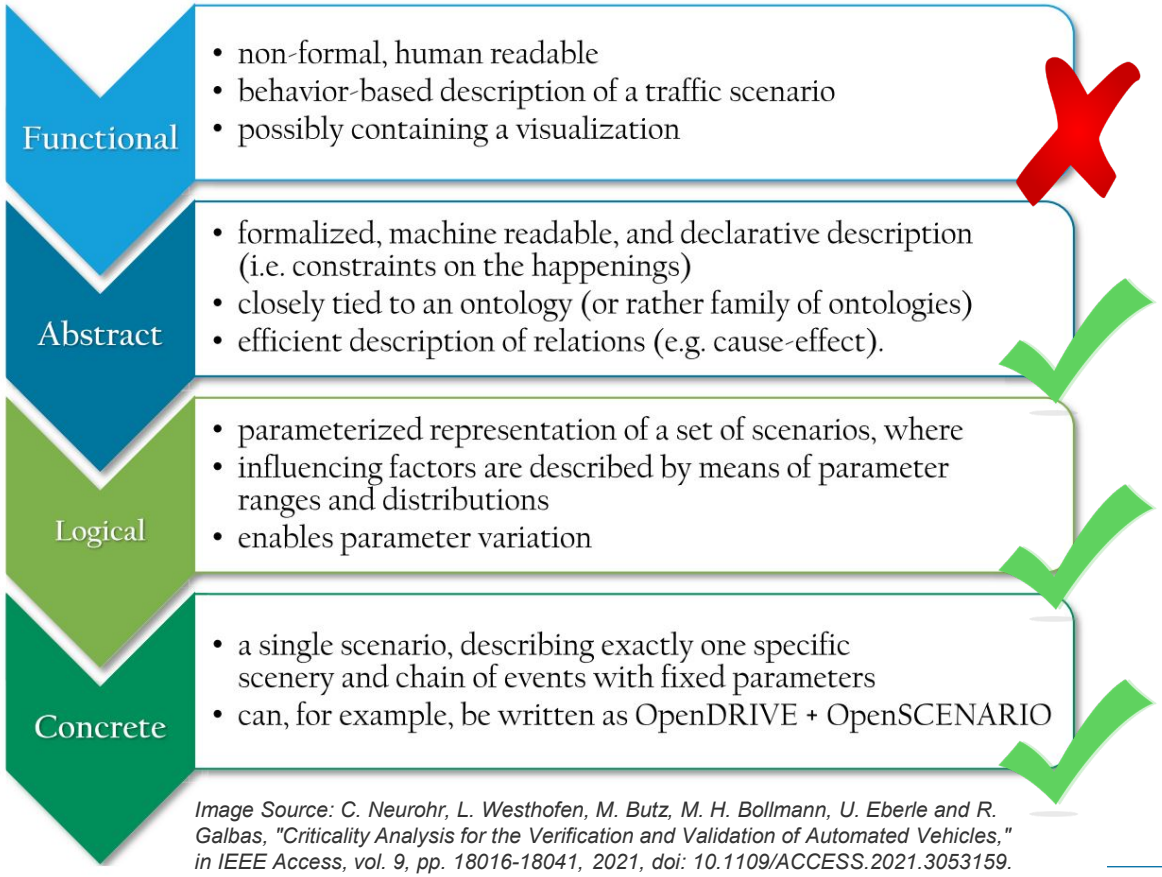


Image Source: C. Neurohr, L. Westhofen, M. Butz, M. H. Bollmann, U. Eberle and R. Galbas, "Criticality Analysis for the Verification and Validation of Automated Vehicles," in IEEE Access, vol. 9, pp. 18016-18041, 2021, doi: 10.1109/ACCESS.2021.3053159.

- Clarification - This is a spectrum not a set of discrete levels
- Concrete scenarios
  - How concrete is a concrete scenario? (concrete vs executable)
  - Two valid options:
    a) A scenario is concrete if everything an author is interested in observing is defined
    b) A user wants full control of a scenario and assigns all parameters values
- Abstract scenarios
  - Scenario A is an abstract form of scenario B if its legal scenario space is a superset of that of scenario B

An ASAM OpenSCENARIO 2.0 Scenario

Scenario

- An OpenSCENARIO 2 scenario is a description of the behaviour or temporal evolution of physical objects and environmental conditions on the driving infrastructure over an interval of time, including the movement of traffic participants or the change of environmental conditions.

- A full scenario description provides all information neccessary to determine
  - where it takes place - driving infrastructure - road layout, road furniture and other static objects
  - who is involved - Actors (vehicles, objects, traffic lights, pedestrians, etc.) & environmental conditions
  - when do they do what – Actions triggered by events or temporal operators

- A scenario may include
  - a specification of validity criteria..

- A scenario may refer to simulations, real tests, driving data, or any combination thereof.

- N:N relationship between a scenario and an OSC2 file.

Language Basics

- Statically typed language
- Parameter and variable types are known at compile time
- For now, only a small set of basic types is defined
- Physical types and units adopt the approach of FMI, with an exponent base in SI units that can be combined
- Standard will include a structured EBNF grammar
- More detail on the basic constructs in the webinars!
  - Compound types
  - Methods
  - Inheritance (conditional & unconditional)
  - Parameter vs. variable
  - Etc.

Object Pathing

- An OSC2 map consists of two types of objects
  1. Instances of junction, which connect roads in the road network
     - Contains a list of connecting road references
  2. Instances of route, the parent class for map elements, like road, lane, etc.

- A route denotes something an object can move on, the along() modifier assigns this to e.g. a vehicle.
  - This object can be used both for very concrete and very abstract definitions

```
r1: route
car1.drive() with:
   along(r1)
   speed([5..10]kph)
```

  - A route element can be any of road, lane, lane section, junction_road, path (points along a road), or a free space path (points not directly on road infrastructure) or any combination thereof (compound_route)
- Additionally, trajectory elements can be used to define a path with a time dimension

```
t1: trajectory = files.read_trajectory("trajectory1.json") do car1.drive() with: along_trajectory(t1)
```

Object Pathing

- An OSC2 map consists of two types of objects

  1. Instances of junction, which connect roads in the road network
     - Contains a list of connecting road references

  2. Instances of route, the parent class for map elements, like road, lane, etc.

- A route denotes something an object can move on, the along() modifier assigns this to e.g. a vehicle.
  - This object can be used both for very concrete and very abstract definitions

```
r1: route
car1.drive() with:
   along(r1)
   speed([5..10]kph)
```

  - A route element can be any of road, lane, lane section, junction_road, path (points along a road), or a free space path (points not directly on road infrastructure) or any combination thereof (compound_route)

- Additionally, trajectory elements can be used to define a path with a time dimension

```
t1: trajectory = files.read_trajectory("trajectory1.json") do car1.drive() with: along_trajectory(t1)
```

Object Pathing

- OSC2 aims to enable both very concrete references, e.g. loading an OpenDRIVE map then creating a route

```
map: map = “straight_road.xodr"
r: route = map.route_from_points([map.odr_point(…),
map.odr_point(…), …])
```

- Or using its abstract road syntax without a concrete map to merely specify constraints for a road network that allow an abstract definition without dependency on a specific map.
- Note that 2.0 is likely to only contain a minimal featureset of the full syntax

```
r1: route with:
   keep(length > 100m)
   keep(it.min_lanes == 2)
```

- Another feature still under discussion is the ability to create routes from named roads

```
create_route_for_named_road(“Arcisstraße”)
```

Actions

- A fundamental, non-decomposable behavior of an actor
- Used whenever the state of an actor is expected to change
- Platform and implementation agnostic – virtual & physical platforms
- An actor may undergo multiple actions simultaneously
- Temporal operators determine the order of actions
- Events trigger the start & end of actions
- Two types of actions
  - Generic actions – An action with a rich set of controls on how to perform it
  - Specialized actions - A commonly used set of actions to support reuse and readability
- Both action types can be mixed in the same scenario

Actions

Generic Action

- More “wordy”
  → easier to tune/modify for author –specific needs

```
scenario generic_action:
        duration: 50s
    do vehicle1.drive() with:
            speed(5kph, at: end)
            acceleration(5kps)
```

Specialized Action

- More concise & readable
  → Author’s intent is more clear

```
scenario specialized_action:
    do vehicle1.reach_speed(5kph)
```

```
scenario vehicle.reach_speed:
    speed: speed
    do drive() with:
        speed(speed, at: end)
```

Modifiers

- Language constructs that modify the behavior of the scenarios in which they are nested

- **Basic modifiers**
  - built into the language
  - keep(), cover() and record() – universal modifiers (can appear in any object)
  - on, until, in – scenario-only modifiers

```
ego: car(category: car):
   keep (width == 2.0m)
   keep (bounding_box.center_x == 1.4)
```

- **Compound modifiers**
  - defined using the language and contain other scenario members
  - Only usable in scenarios

```
car1.drive() with:
    speed(value: [5..7]kph, faster_than: car2, at: end)
```

```
modifier car.speed of drive:
    value: speed
    faster_than: car
    slower_than: car
    at: at_kind
    keep(faster_than == null or slower_than == null)
```

Constraints

- Define the allowed values of parameters

```
ego_speed: speed              # parameter type declaration
keep (ego_speed > 70kph)      # constraint
```

- Three types of constraints
  - **Hard** – MUST be adhered to by parameters
  - **Soft** – a recommendation
  - **Default** – a default value that can be overridden explicitly

- Shorthand for parameter assignments in scenarios

```
do foo(x: 7)      ==      do foo() with:
                              keep(it.x == 7)
```

Temporal Operators

- The “when” in OSC2 is defined via two language elements:
  1. Composition operators
  2. Events

- OSC2 composition operators such as serial, parallel, one_of allow users to construct phases or temporal labels for when a scenario invocation or action instantiation occurs.
  - one_of models the occurrence of one scenario from a set of two or more scenarios
  - parallel models overlapping occurrences of a primary scenario with one or more secondary scenarios (parallel, overlap@start, initial, any, etc.)

- OSC2 Events resolve to a specific point in time within the scenario. This allows users to:
  - Resolve the start and/or end of a phase.
  - Resolve a moment to take a measurement in the scenario.

Events

- A zero time occurrence in time, optionally with parameters

```
event car_arrived  # A single event declaration
```

- Qualified events (QE) are events with expressions that evaluate to true or false at any given time

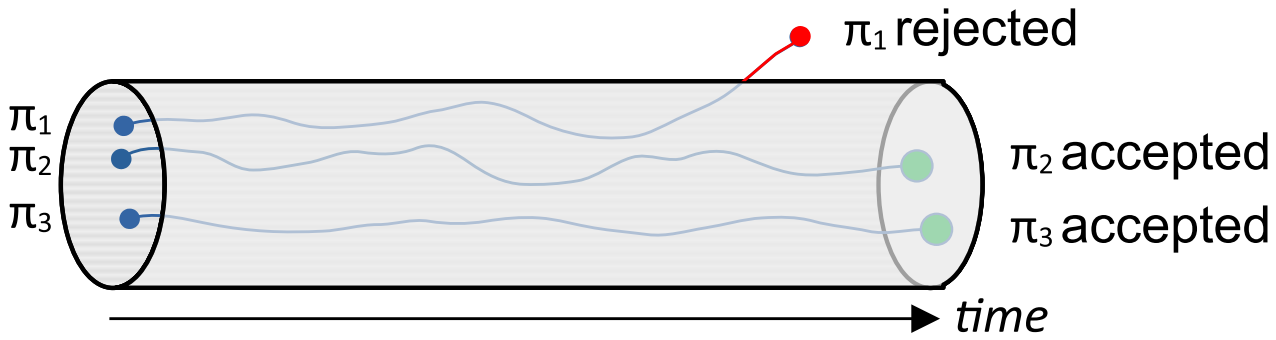
```
[Boolean-exp][@event-path]
```

- A QE can be used with modifiers and actions:

```
wait distance_to(car2) < 10m # Wait until the QE is true
until (@car_arrived)         # Terminate the scenario when the QE is true
```

Semantics of Scenario Concretization

- A *trace*, π, is a behavior occurring in a traffic system, or a model thereof, such as a simulation.
- A set of scenarios defines a set of traces that are accepted by the OSC2 model
- A trace is rejected if the behavior is out of scope of the OSC2 model
- This has no relation to passing/failing of tests → Test criteria are only evaluated for accepted traces




```
scenario my_scenario:
  do actor.drive() with:
      speed(speed: 0kph, at: start)
      speed(speed: 10kph, at: end)
```

Trace is only accepted if start speed is 0kph and end speed is 10kph

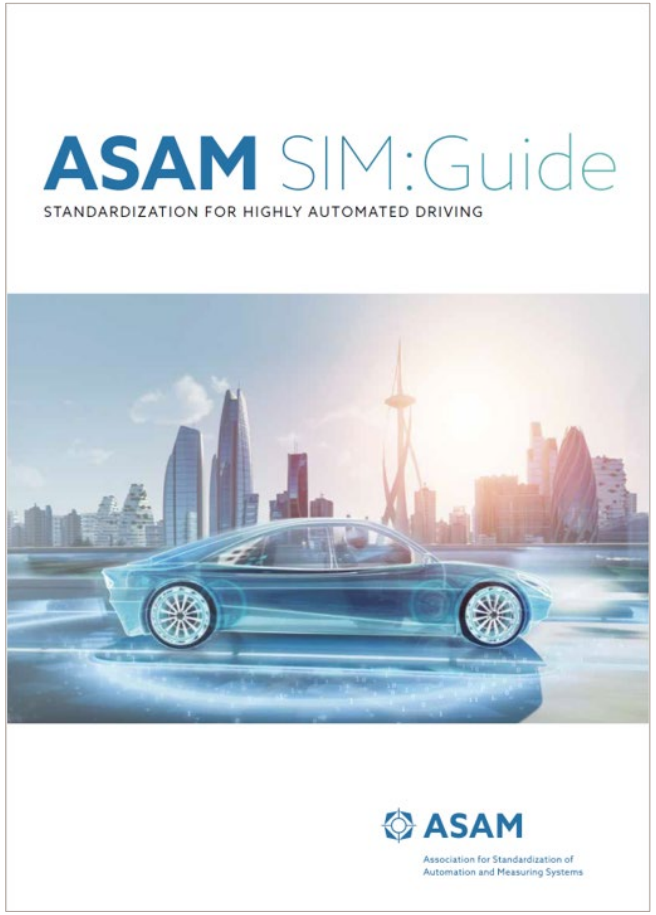
ASAM OpenSCENARIO 2.0

Closing Words

- It is not too late to join in!
- Review open to ASAM membership in October – we need your feedback!
- OpenSCENARIO Implementers Forum is ongoing
  → If you want a jumpstart on figuring out how to use and integrate the language, join us!
- Release to the public in December
- This is just the beginning… 2.1 development to start early next year!
- More details on in-depth webinars coming soon!
- Please reach out to myself (benjamin.engel@asam.net) and/or the project lead, Gil Amid (gil.amid@foretellix.com) for any questions or comments!

ASAM SIM:Guide

Standardization for Highly Automated Driving



The guide contains

- Introduction to the ASAM domain "Simulation"
- All current standardization activities by ASAM in the domain
- Current standardization activities outside of ASAM
  -> places the ASAM OpenX standards in the larger context of a global standardization landscape
- Application stories from our members
  -> how do our members use the OpenX standards
  -> how have these standards helped to improve processes
  -> how are they facilitating customer projects

Order your copy today:
www.asam.net/asam-guide-simulation

Benjamin Engel & Nicco Hagedorn
Global Technology Managers
ASAM e.V.