# OpenSCENARIO 2.0.0

## ASAM Technical Seminar

**Rolf Magnus**
**Virtual Validation Expert, Akka Group**

April 27, 2022

ASAM Association for Standardization of Automation and Measuring Systems

# Agenda

| | |
|---|---|
| 1 | Current Status |
| 2 | Elements of OpenSCENARIO 2.0.0 and Abstraction Levels |
| 3 | Scenario Composition |
| 4 | Implementations |

# OpenSCENARIO 2.0.0
## Current Status

ASAM

# Current Status / Public Release Candidate

An OpenSCENARIO 2.0.0 Public Release Candidate was released end of last year

Goals:

- Give people the opportunity to start using OpenSCENARIO 2
- Review the standard to work out remaining issues
- Clarify details further
- Gather feedback from public
- Improve document structure

Non-Goals:

- No major design changes or entirely new features

- Three main groups – **Language**, **Domain Model** and **Implementers Forum**
- Release Candidate phase was started with a public Implementers Forum kickoff in January
- Implementers Forum closed by end of March
- Hackathon in April

Result: Refined standard
Status: Standard is mostly ready, now waiting for final approval by the Technical Steering Committee in May

# OSC2 Implementers Forum

Opportunity for the public to:

- Ask the standard writers questions / get clarification
- Report back inconistencies or ambiguities in the standard
- Implement scenarios to learn how to apply the standard to real test cases
  - UNECE 157 – ALKS swerving side vehicle test
  - EURO NCAP CPNC-50 – Automatic emergency braking
  - EURO NCAP LSS 3.0.2 – Emergency lane keeping / overtaking test
  - V&VM FUC 2.3 – Urban junction test for AD vehicles with emergency brake

→ The Implementers Forum was a great source for feedback and information helping to improve the standard

# Elements of OpenSCENARIO and Abstraction Levels

# Abstraction Levels in OpenSCENARIO V2.0.0

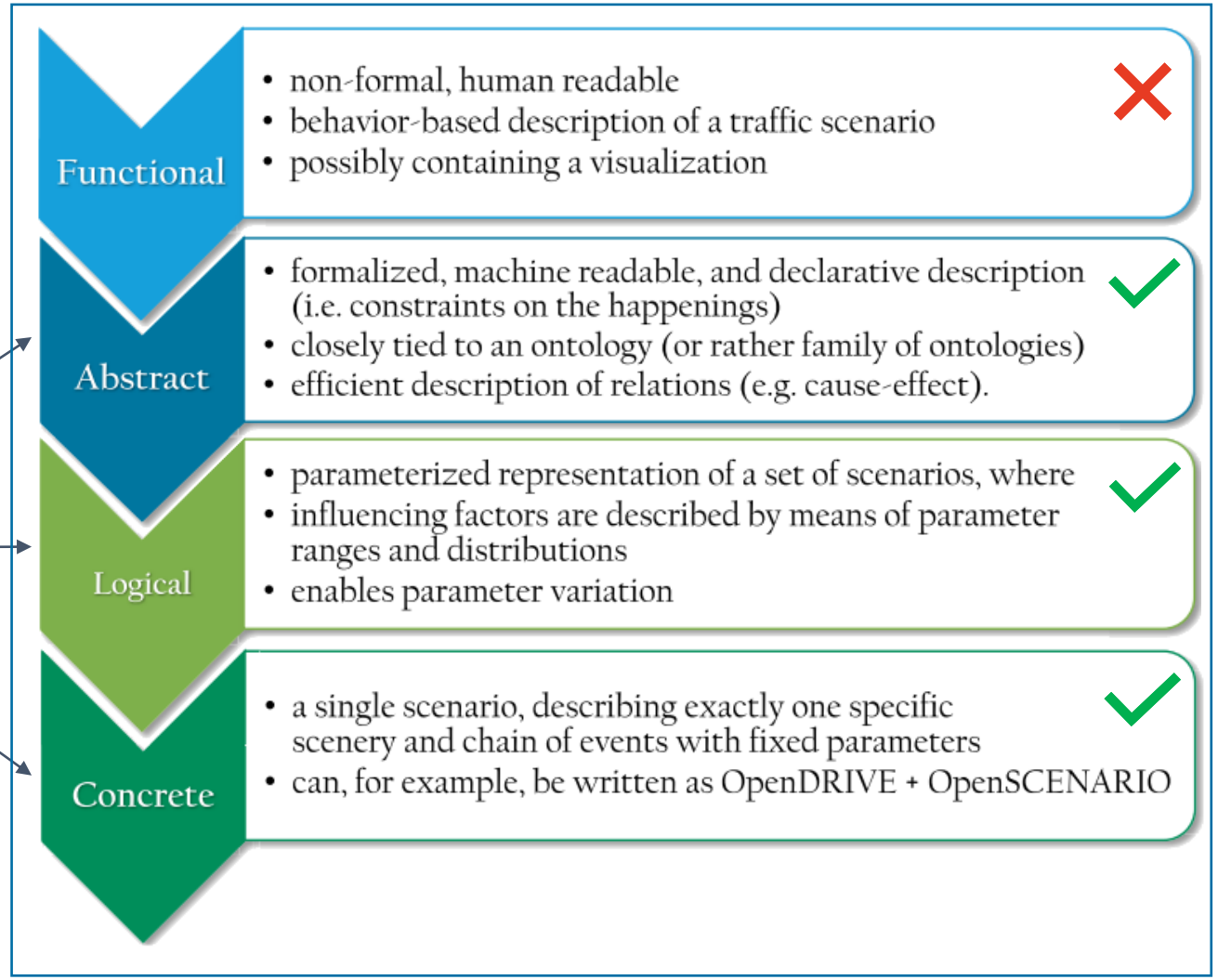
Not in the scope of OSC 2.0

→ Manual conversion to another level required

Can be stored in OpenSCENARIO

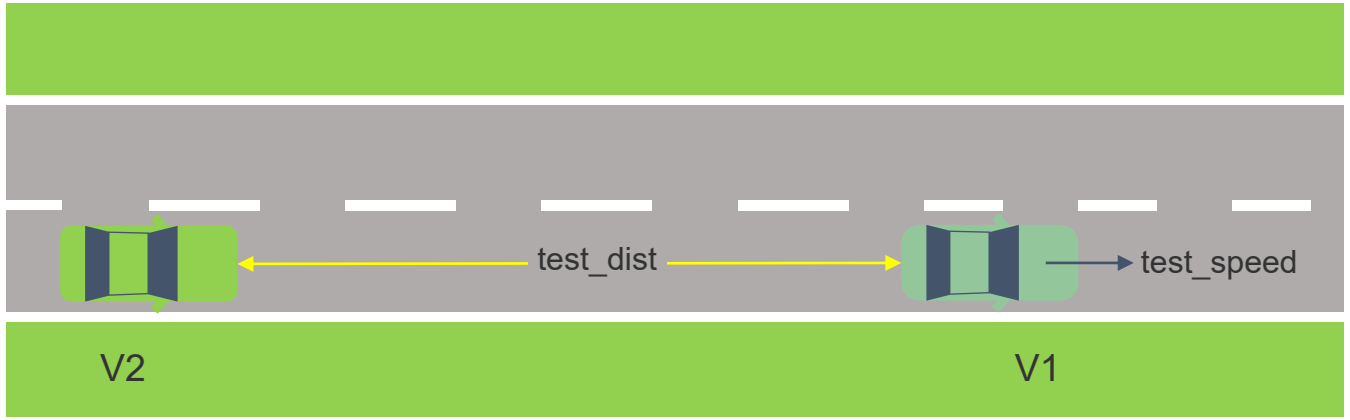
→ Automation of conversion is possible

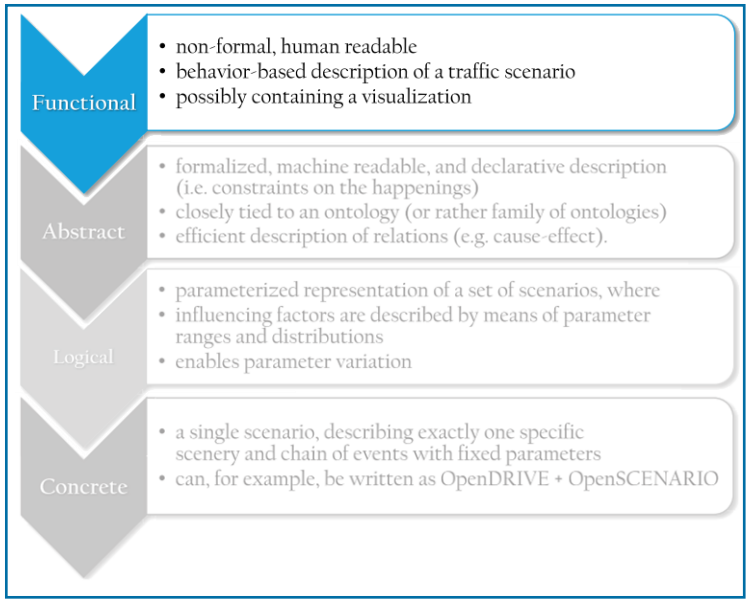
**One standardized format for all three levels**



# Abstraction Levels in OpenSCENARIO V2.0.0

- The scenario describes two vehicles, v1 and v2
- v1 is driving at a constant speed (test_speed)
- v2 is following v1 at a certain distance (test_dist)



# Abstraction Levels in OpenSCENARIO V2.0.0

```
# Simple scenario where the Ego vehicle follows
# another vehicle at a constant distance
import osc.standard

scenario follow:
    v1, v2:     vehicle

    test_speed: speed
    test_dist:  length

    do parallel(overlap: equal):

        v1.drive() with:
            keep_lane()
            speed(speed: test_speed)

        serial:
            v2.change_space_gap(reference: v1,
                                direction: behind
                                target: test_dist) with:
                lane(same_as: v1)
            v2.keep_space_gap(reference: v1,
                              direction: longitudinal)
```

standard library import

scenario declaration

actor parameters

physical type parameters

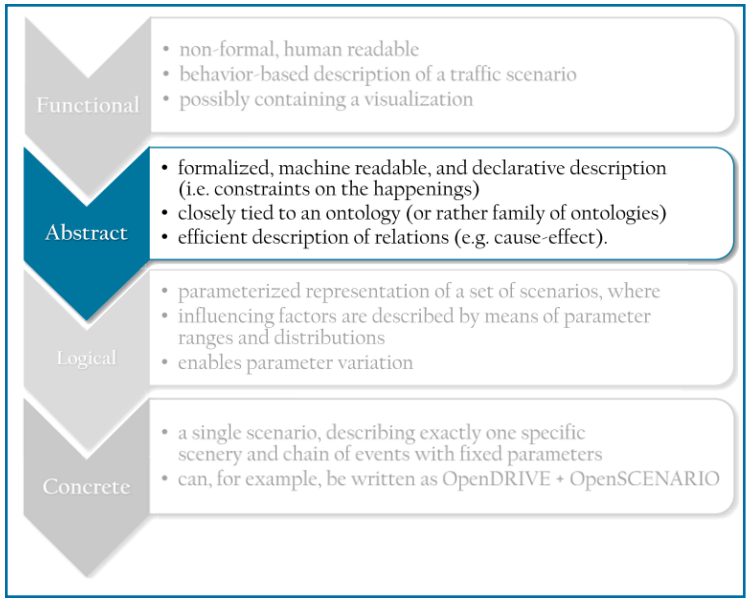
behavior

scenario composition operator

generic action

modifiers

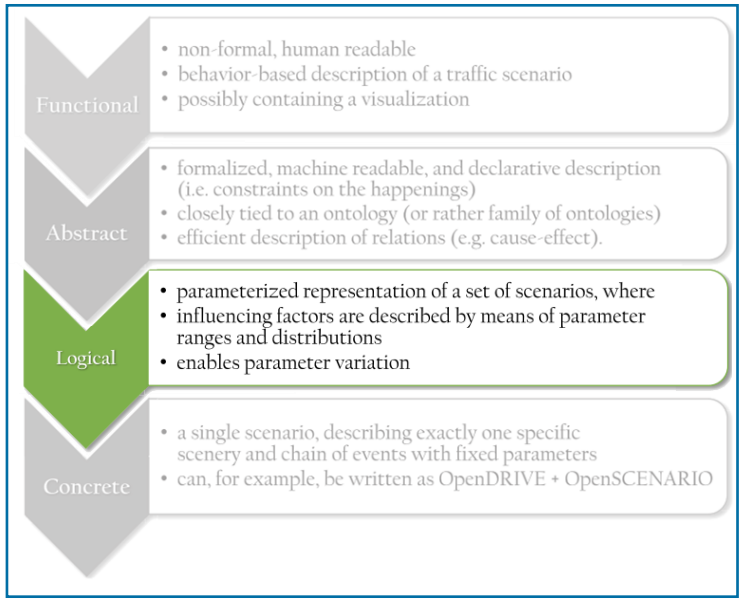
specialized action



# Abstraction Levels in OpenSCENARIO V2.0.0

```
# Simple scenario where the Ego vehicle follows
# another vehicle at a constant distance
import osc.standard

scenario follow:
    v1, v2:     vehicle

    test_speed: speed
    test_dist:  length

    keep(test_dist in [30m .. 50m])
    keep(test_speed in [40kph .. 60kph])

    do parallel(overlap: equal):

        v1.drive() with:
            keep_lane()
            speed(speed: test_speed)

        serial:
            v2.change_space_gap(reference: v1,
                                direction: behind
                                target: test_dist) with:
                lane(same_as: v1)
            v2.keep_space_gap(reference: v1,
                              direction: longitudinal)
```

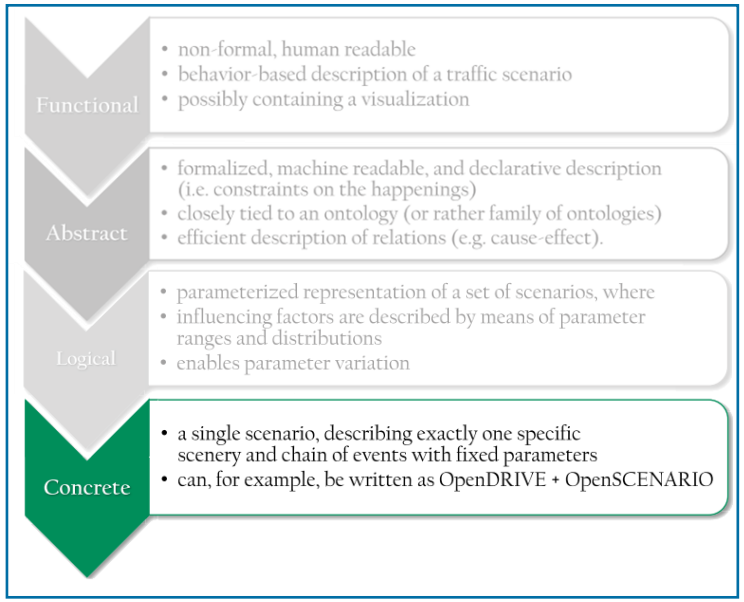
constraints



# Abstraction Levels in OpenSCENARIO V2.0.0

```
# Simple scenario where the Ego vehicle follows
# another vehicle at a constant distance
import osc.standard

scenario follow:
    v1, v2:     vehicle

    test_speed: speed
    test_dist:  length

    keep(test_dist == 34m)
    keep(test_speed == 55kph)

    do parallel(overlap: equal):

        v1.drive() with:
            keep_lane()
            speed(speed: test_speed)

        serial:
            v2.change_space_gap(reference: v1,
                                direction: behind
                                target: test_dist) with:
                lane(same_as: v1)
            v2.keep_space_gap(reference: v1,
                              direction: longitudinal)
```

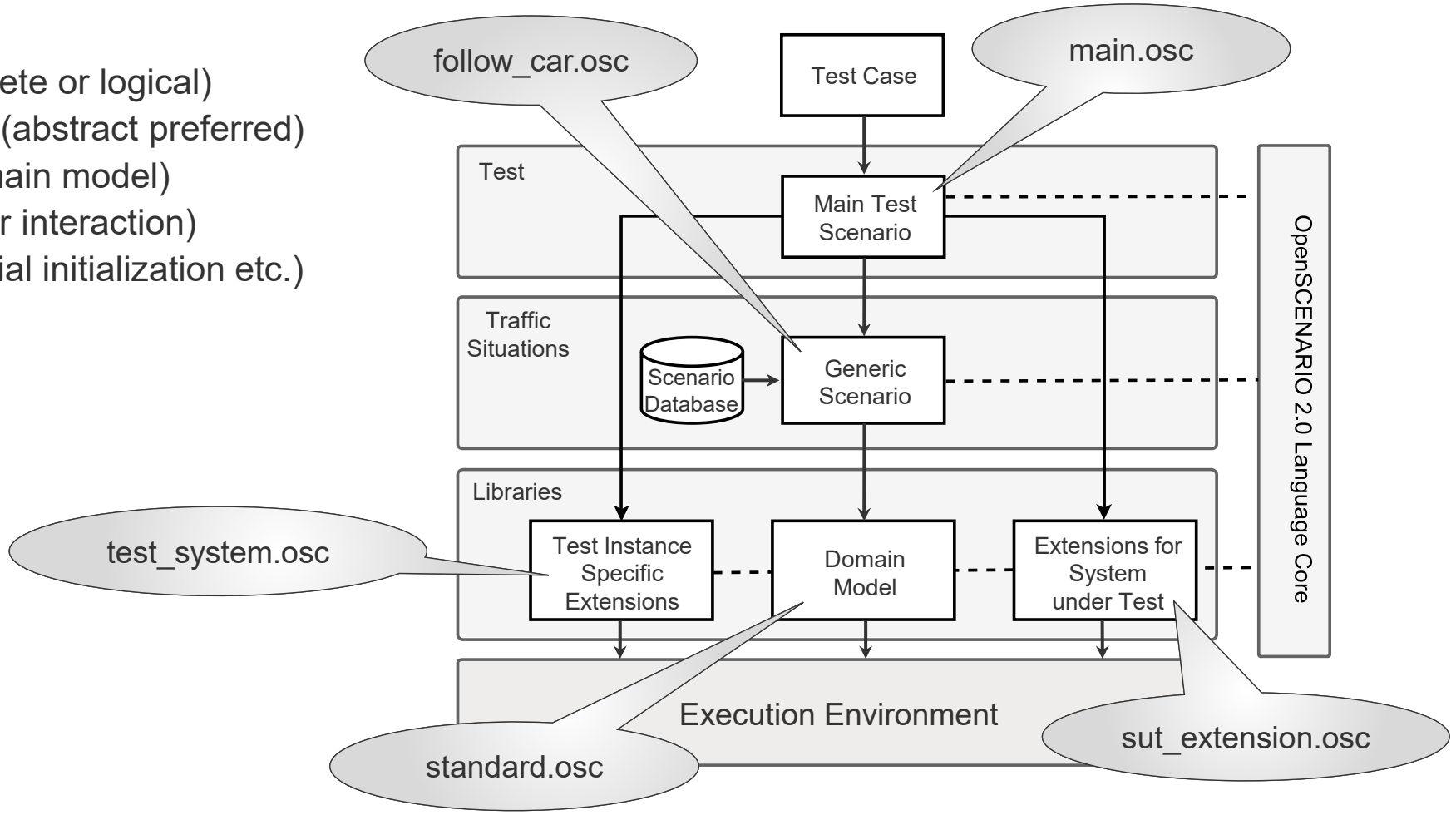
constraints



# Scenario Composition

# Scenario Composition

- Test case
- Main test scenario (can be concrete or logical)
- Generic scenario from database (abstract preferred)
- Standard library (represents domain model)
- SuT specific extensions (like user interaction)
- Extensions for test system (special initialization etc.)



# Implementations

# OpenSCENARIO2.0 Scenario Compiler

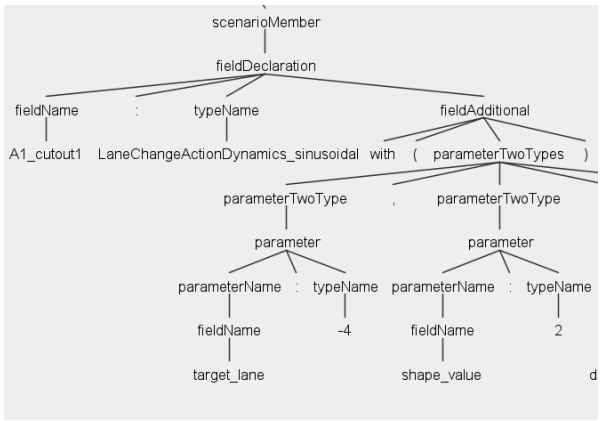
Preprocess | Compiler Frontend - Antlr4 | Compiler Backend - Scenario Engine

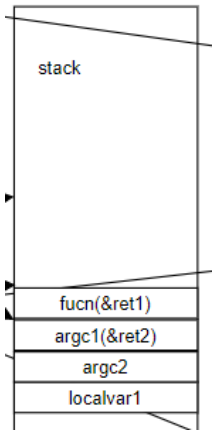
Standard Library | DSL Script | Syntax Tree | Script Engine | Semantic Model

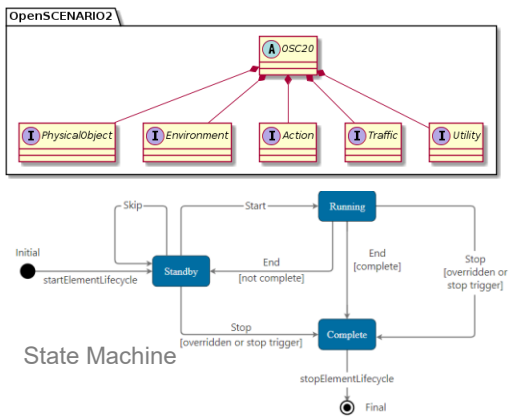
Standard Library | Grammar - EBNF | Lexical & Syntax Parser | Semantic Parser | Domain Model

```
type distance SI(m: 1)
type speed SI(m: 1, s: -1)

actor car_ego: Vehicle:
    driving_style:DrivingStyle

actor car: Vehicle:
    driving_style:DrivingStyle

def normal(mean: real, sd: real):r
    external.python("numpy.random.
```

```
grammar Scenario;
// Parser rules
osc
    :   preludeStatement* oscStatement* EOF
    ;

preludeStatement
    :   importStatement
    ;

oscStatement
    :   structDeclaration
    |   actorDeclaration
    |   actionDeclaration
    |   modifierDeclaration
    |   typeDeclaration
    |   unitDeclaration
```

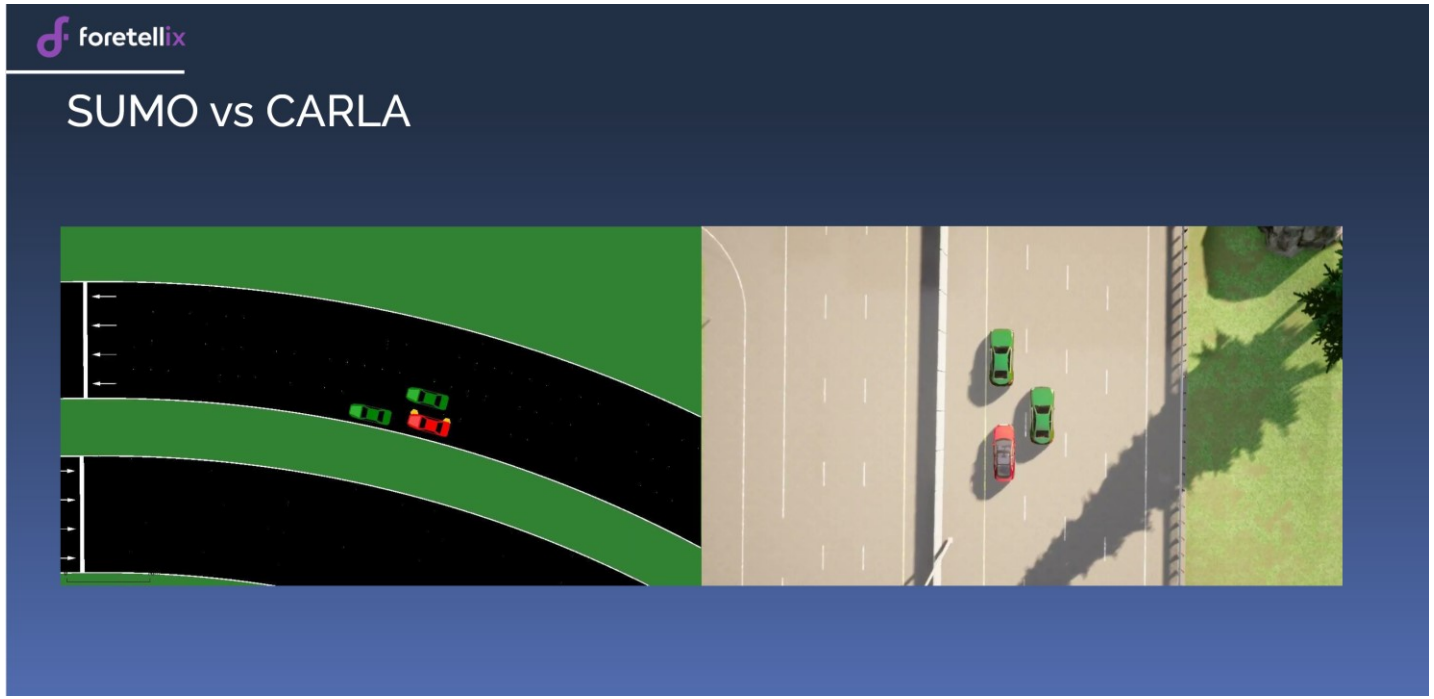



State Machine

# Foretellix OpenSCENARIO Execution




## ASAM OpenSCENARIO 2.0

The ASAM standard is revolutionizing the way safety is being developed into automated driving systems.

Foretellix is the pioneer and expert in OSC2.0, leading the standard development project and being the first to offer a native implementation in its Foretify ™ platform.


### OSC 2.0 Native Implementation

The only build to standard implementation build using abstract scenarios, automation & coverage analytics

### OSC 2.0 Expertise & Methodology Leadership

Foretellix's technology, methodology, and experts have led the standard development

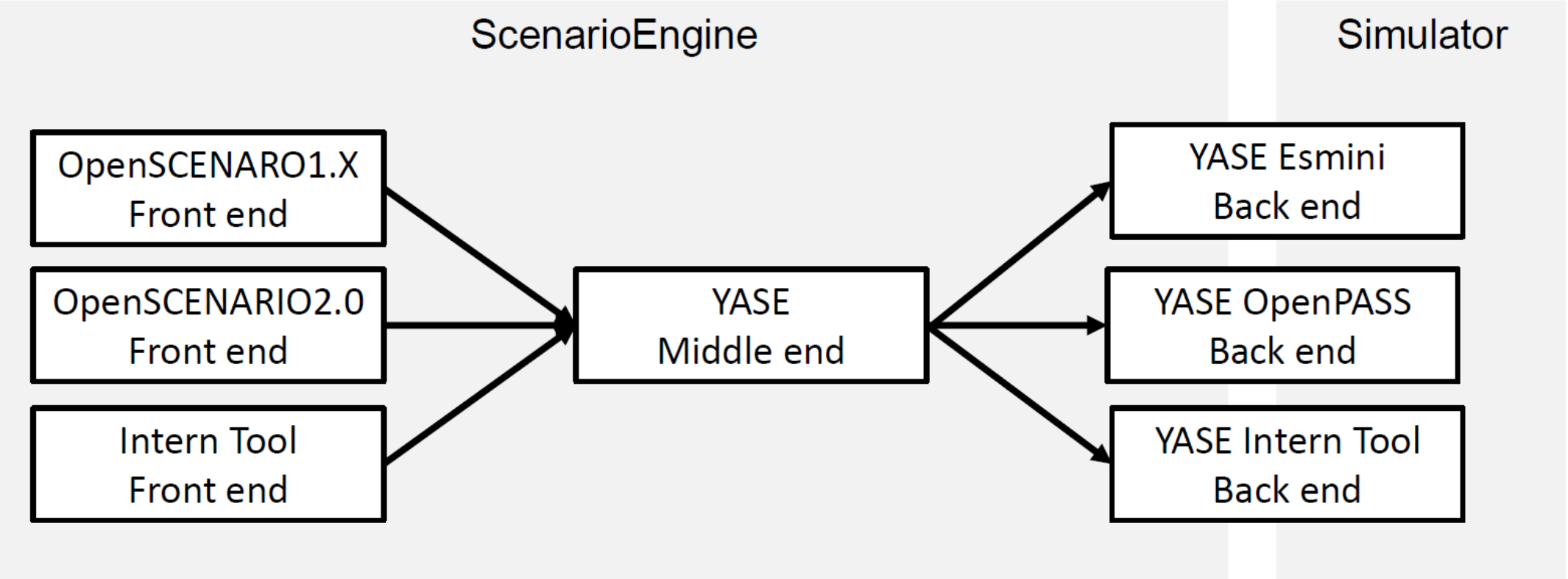
### OSC 2.0 V-suite Packages

Ready-made content packages for L2 ADAS, L3 and L4 to get you started in no time

# YASE (Bosch)

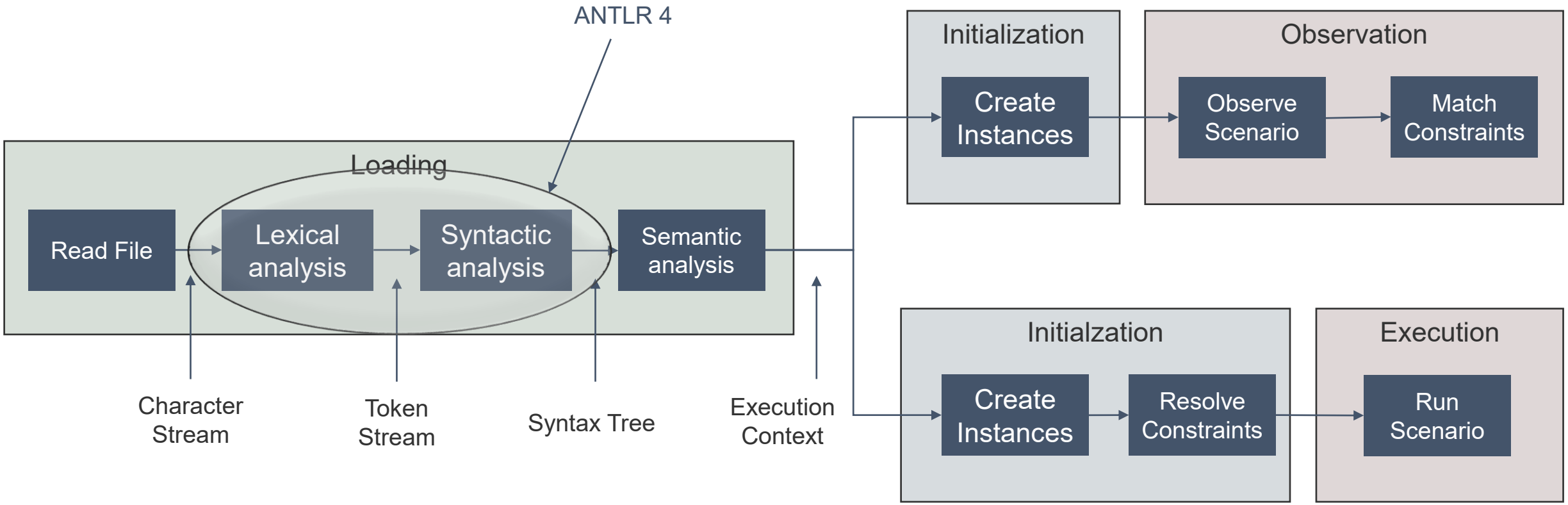
## Three phases of scenario compiler/ ScenarioEngine

SET Level



# Akka Implementation

## Scenario Execution / Observation Steps



# Syntactic and Semantic Checking Example

```
1  import osc.standard
2
3  scenario follow:
4  v1, v2:     vehicle
5
6  test_speed: speed
7  test_dist:  length
8
9  keep(test_dis in [30m .. 50s])
10 keep(test_speed in [40kph .. 60kph])
11
12 do parallel(overlap: equal):
13
14     v1.walk() with:
15         keep_lane()
16         speed(test_speed)
17
18     serial:
19         v2.change_space_gap(reference: v1, direction: behind, target: test_dist) with:
20             lane(same_as: v1)
21         v3.keep_space_gap(reference: v1, direction: longitudinal)
```

```
Error:   oscfiles/osc2/follow_broken.osc:14:8: behavior does not exist
                 v1.walk() with:
                    ↑--↑
Error:   oscfiles/osc2/follow_broken.osc:21:12: actor instance 'v3' does not exist
                     v3.keep_space_gap(reference: v1, direction: longitudinal)
                     ↑↑
Error:   oscfiles/osc2/follow_broken.osc:9:9: could not resolve identifier 'test_dis'
           keep(test_dis in [30m .. 50s])
                ↑------↑
Error:   oscfiles/osc2/follow_broken.osc:9:21: no common type found for 'length' and 'time'
           keep(test_dis in [30m .. 50s])
                            ↑----------↑
**** aborted with 4 errors ****
```

# Open Source Tools

51WORLD OpenSCENARIO2 Grammar Checker (osc2checker)

https://github.com/51WORLD/osc2checker

PMSF py-osc2 Framework

https://github.com/PMSFIT/py-osc2

# Thank you for your attention!

Rolf Magnus
Akka Group

rolf.magnus@akka.eu
Phone: +49 151 7463 1734