

ASAM OpenSCENARIO

Quo Vadis ?

Gil Amid
Foretellix Ltd

Andreas Rauschert
BMW Group

Mirko Bulaja
AVL

Benjamin Engel
ASAM e.v.

11. Januar 2024



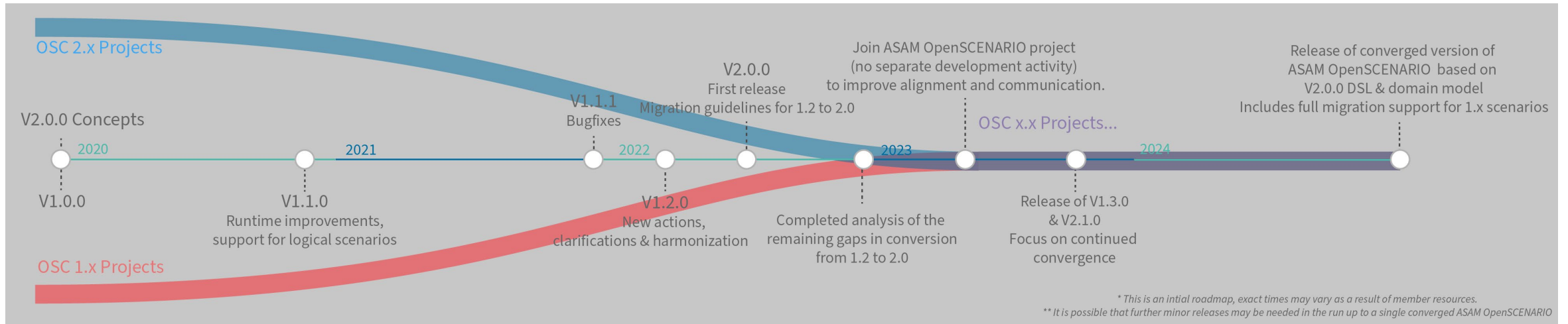
Background

- ASAM OpenSCENARIO 2.0.0 is being released.
- With the release , several clarifications are needed:
 - ASAM roadmap with respect to the two major versions: 1.x and 2.x, and convergence path.
 - Difference between the two major versions (1.x, 2.0.0)
 - Migration path – key points,

Outline

- ASAM roadmap
- Difference between the two major versions (1.x, 2.0.0)
- Migration path – key points,

ASAM OpenSCENARIO ROADMAP



Outline

- ASAM roadmap
- Difference between the two major versions (1.x, 2.0.0)
- Migration path – key points,

ASAM OpenSCENARIO 1.x

- What is OSC1 ?
- OpenSCENARIO 1.x comprises the specification and file schema for the description of the dynamic content in driving simulation applications. The primary use of OpenSCENARIO is the description of complex maneuvers that involve multiple vehicles. OpenSCENARIO 1.x is implemented using XML format and a schema.
- OpenSCENARIO 1.x defines the dynamic content of the (virtual) world (e.g. behavior of traffic participants). Static components (such as the road network) are not part of OpenSCENARIO but can be referenced by the format.
- OpenSCENARIO 1.x defines a data model and a derived file format for the description of scenarios used in driving and traffic simulators, as well as in automotive virtual development, testing and validation. The primary use-case of OpenSCENARIO 1.x is to describe complex, synchronized Maneuvers that involve multiple instances of Entity, like Vehicles, Pedestrians and other traffic participants. The description of a scenario may be based on driver Actions (e.g. performing a lane change) or on instances of Trajectory actions (e.g. derived from a recorded driving Maneuver).

ASAM OpenSCENARIO 1.x - Out of Scope

Test configuration description	The standard neither describes the actual test instance nor its structure.
System under test	The exact description of the system under test, e.g. detailed vehicle configuration, sensor placement, sensor models etc. is not part of OpenSCENARIO. [Sensors are candidates for OpenSCENARIO 1.3]
Test case language	Although including a set of driver input, the standard does not attempt to specify all possible user or system interactions with a vehicle.
Test evaluation	Even though the standard includes the evaluation of conditions for triggering actions, there is no concept for creating test verdicts. A scenario end condition can be defined, but not a verdict.
Driver model	As of version 1.0 The standard does not include Driver or Behavioral Driver Model. Controllers were introduced and can be used.
Vehicle dynamics	Although the standard describes <u>maneuvers</u> in a kinematic way, it also defines a very basic vehicle model, which can be used for more realistic vehicle dynamics simulation. The standard does not include all necessary elements to specify advanced motion dynamics.
Road network	The standard does not include elements to describe roads, other than references to an external road network description. The <u>OpenDRIVE</u> standard can be used for this purpose .
3D environment models	The standard only specifies how to refer to external 3D environment models. Further details, like file format or model structure, are not specified.
Environmental models	The standard incorporates elements to specify the current time and weather information but does not describe how this is to be interpreted by the simulator.

ASAM OpenSCENARIO 1.x - Summary

- OpenSCENARIO describes the dynamic content, including the overall description and coordination of behavior of dynamic entities.
- OpenSCENARIO does not specify the behavior models themselves, nor their handling by the simulation engine, including initialization and setup, runtime interfaces, packaging, etc.
- OpenSCENARIO also does not define the road network or any geometric, visual or physical assets and characteristics used in a simulation. These are instead employed through references to other established formats. Hence, in certain contexts, OpenSCENARIO can be considered as a top-level container. It references other specifications for other relevant parts of the overall scenario.

Asam OpenSCENARIO 2.0.0 - High level – key improvements



- Improved readability – Better readable (vs XML)
- Programming language (vs XML) - object oriented DSL
- Consistent scenario description over levels of abstraction (supports Abstract, Logical and Concrete description)
- Easier and consistent binding to external code/functions/methods for every purpose
- Built in Constraints
- Built in KPI and Coverage measurements
- Composability - completely flexible modularization for reusability of scenarios, unlimited nesting and mixing
- Extendible - the language and domain entities
- Designed to support all test platforms: Sil, Hil, Vil, Mil, Proving Ground

ASAM OpenSCENARIO 2.0 - RECAP

- ASAM OpenSCENARIO 2.0 is the format and mechanism to supply dynamic content and functional behavior to all testing and execution platforms, for all driving scenarios ranging from simple motor-way interactions to long-running, complex inner-city traffic scenarios. In addition, it supports static entities, scenario validity criteria, KPIs and measurement, coverage measurements.
- ASAM OpenSCENARIO 2.0 is not backward compatible with version 1.x. A migration guide with functionality and feature mapping is supplied as part of the documents for the standards.

ASAM OpenSCENARIO 2.0 – In slightly less plain language

- This is a declarative domain specific programming language combined with a domain model, specifying entities with their properties
- The language supports abstract, logical and concrete levels of abstraction

	Concrete	Logical	Abstract	Functional
OpenSCENARIO 1 	✓	✓		
OpenSCENARIO 2 	✓	✓	✓	

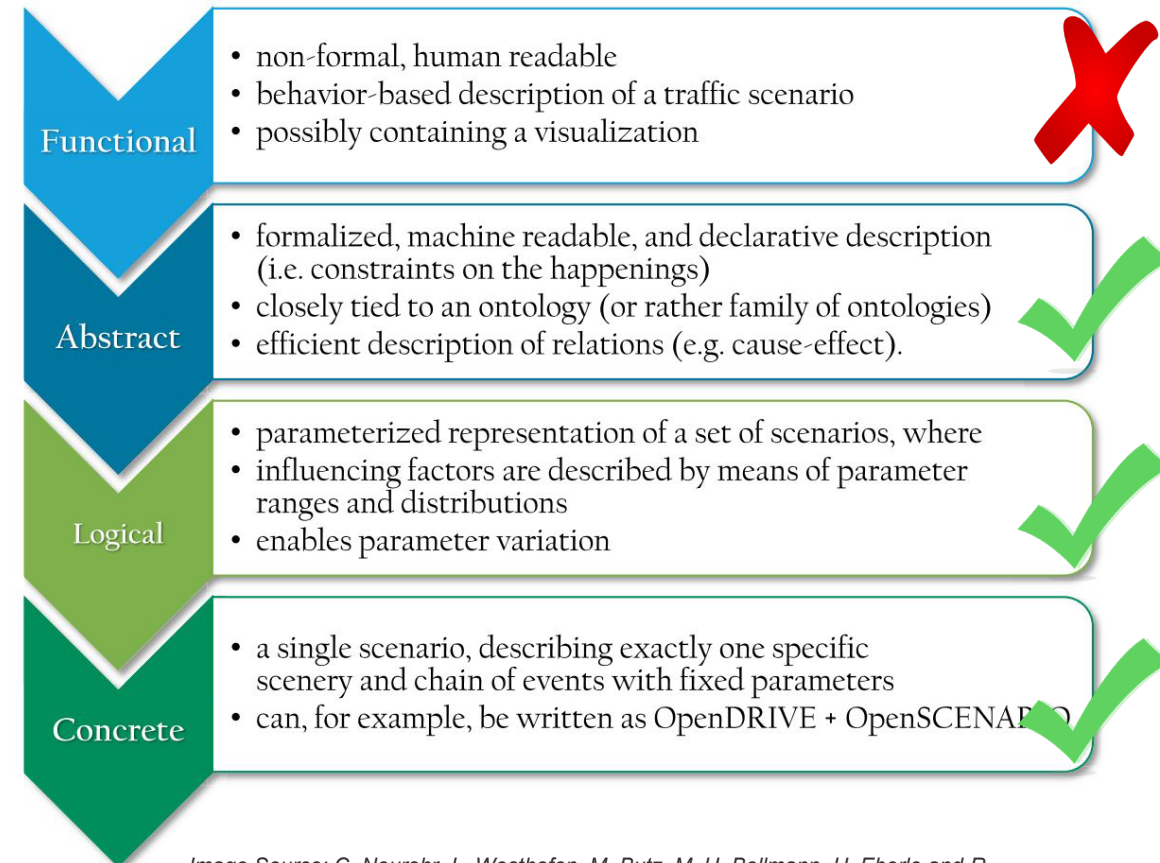


Image Source: C. Neurohr, L. Westhofen, M. Butz, M. H. Bollmann, U. Eberle and R. Galbas, "Criticality Analysis for the Verification and Validation of Automated Vehicles," in IEEE Access, vol. 9, pp. 18016-18041, 2021, doi: 10.1109/ACCESS.2021.3053159.

ASAM OpenSCENARIO 2.0 – In Plain language

The language enables:

- Specification of the temporal order of actions, through language mechanisms such as serial and parallel
- Reuse or combine individual scenarios in order to create more complex scenarios.
- Definition of events, e.g. for data capture or triggering of further actions
- Use events to monitor ADS errors.
- Specification of ranges or any other behavioral/numerical considerations, through constraints
- Setting coverage goals and accumulating coverage
- Abstract road network descriptions that remove dependency on specific maps, allowing one to match the specified constraints to multiple maps/ODDs
- Extending the domain model, adding objects, entities, types and more.

ASAM OpenSCENARIO 2.0.0 – Enabling new capabilities

- The language features enable:
 - Developments and maintenance of test scenarios that are independent of specific map and geography (abstract road networks). Can automatically be mapped to a geography.
 - Consistent reporting and documentation of error checking and scenario test coverage results – using built in checking mechanism and coverage accumulation.
 - Shared and consistent definition of ADS errors, thresholds and pass/fail criteria – using constraints, events and built-in checking.
 - Consistent definition of required testing and coverage ranges for every value/parameter (from colors of the vehicles to acceleration
 - Reuse or combine individual scenarios from libraries/catalogs in order to create more complex scenarios.

ASAM OpenSCENARIO 1 – 2 levels of abstraction

Concrete: scenario.xosc

```
<!-- scenario.xosc -->
<ParameterDeclarations>
  <ParameterDeclaration name="Ego_speed"
                        parameterType="double" value="10"/>
</ParameterDeclarations>
...
<Actions>
  <Private entityRef="Ego">
    <PrivateAction>
      <LongitudinalAction>
        <SpeedAction>
          <SpeedActionDynamics dynamicsShape="step"
                              dynamicsDimension="time" value="0"/>
          <SpeedActionTarget>
            <AbsoluteTargetSpeed value="$Ego_speed"/>
          </SpeedActionTarget>
        </SpeedAction>
      </LongitudinalAction>
    </PrivateAction>
  </Private>
</Actions>
```

Logical: scenario.xosc + distribution.xosc

```
<!-- distribution.xosc -->
<ParameterValueDistribution>
  <ScenarioFile filepath="scenario.xosc"/>
  <Deterministic>
    <DeterministicSingleParameterDistribution
      parameterName="Ego_speed">
      <DistributionRange stepWidth="10">
        <Range lowerLimit="10" upperLimit="60"/>
      </DistributionRange>
    </DeterministicSingleParameterDistribution>
  </Deterministic>
</ParameterValueDistribution>
```

ASAM OpenSCENARIO 2 – 3 levels of abstraction

Concrete – specific test

```
ego.drive() with:  
  speed(10kph)
```

Logical (R-157 speed range)

```
ego.drive() with:  
  speed([10kph..60kph])
```

Abstract

```
ego.drive() with:  
  keep(it.speed <= speed_limit)
```

“The ADS shall not cross the legal speed limit “


ASAM OpenSCENARIO 2.0 – Examples

Example for use of the parallel operator

```
scenario parallel_phases:  
  v1, v2: vehicle  
  do parallel():  
    phaseA: v1.drive() with:  
      speed(speed: 0kph, at: start)  
      speed(speed: 10kph, at: end)  
    phaseB: v2.drive() with:  
      speed(speed: [10..15]kph)  
      position(distance: [5..100]meter, behind: v1, at: start)
```

Example for emit event

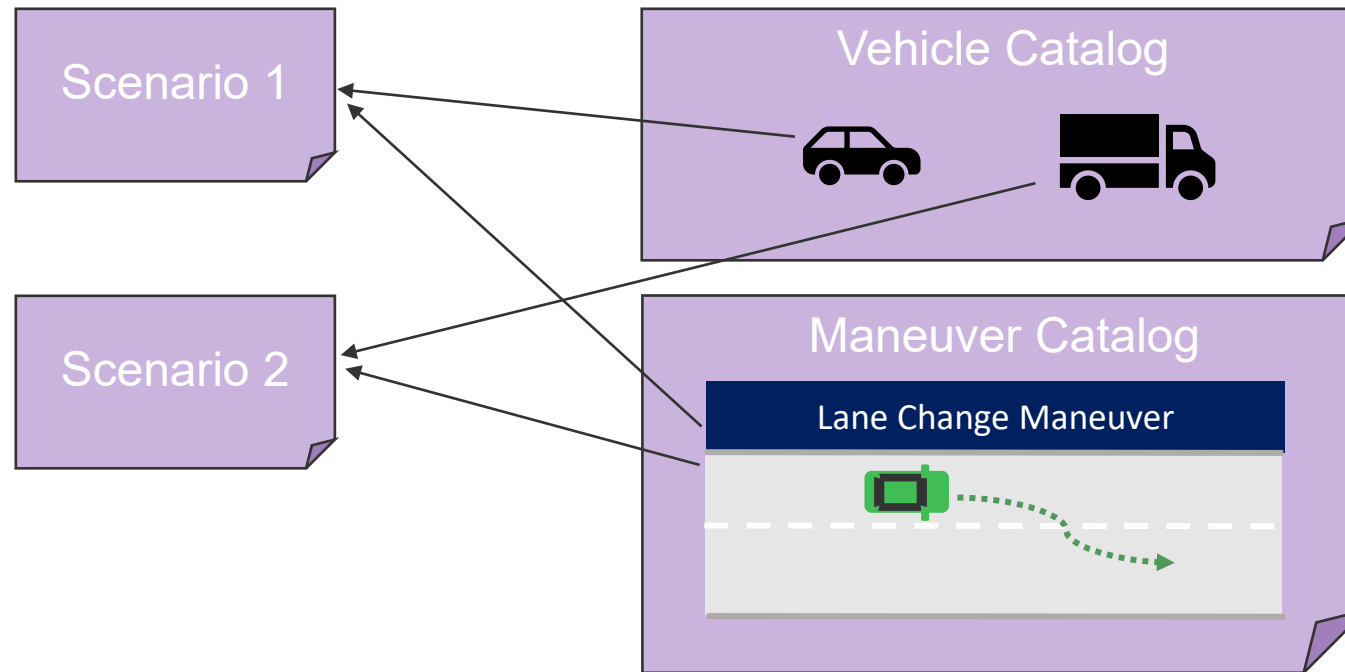
```
event vehicle_reached_speed(vehicle : vehicle, vehicle_speed : speed) # event definition  
  
...  
  
scenario car.two_phases:  
  do serial:  
    phase1: drive() with:  
      speed(speed: 0kph, at: start)  
      speed(speed: [25..35]kph, at: end)  
    emit vehicle_reached_speed(actor, actor.speed)  
    phase2: drive() with:  
      speed(speed: [30..50]kph)
```



ASAM OpenSCENARIO 1 – Scenario Composition

Use catalogs to reuse scenario sections:

- Only one hierarchy level
- Only specific predefined catalogs



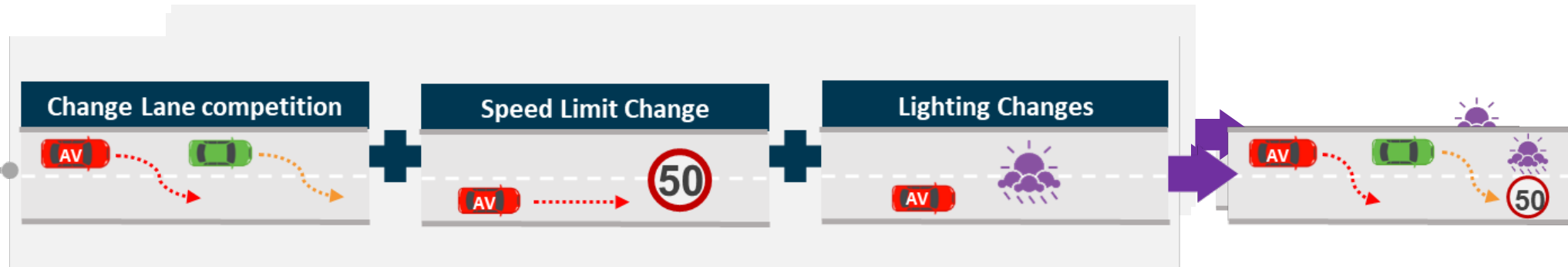
ASAM OpenSCENARIO 2 - Scenario Composition



Easily compose complex scenarios, by combining simple scenarios in serial or parallel

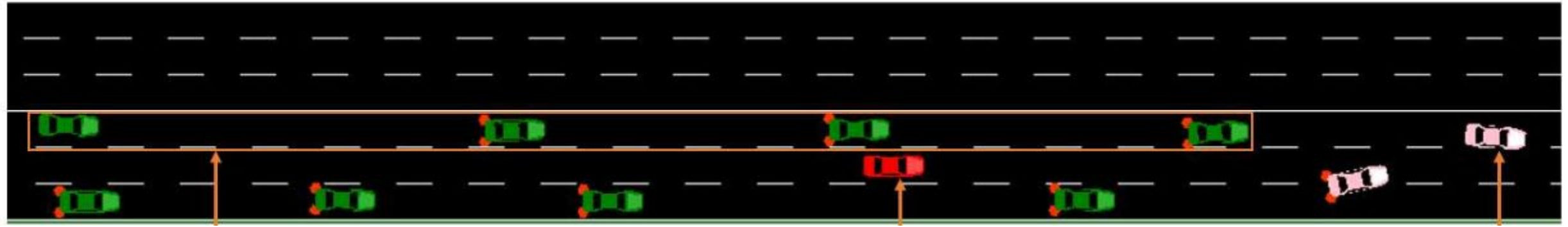
Use catalogs/libraries to create complex interactions and ODD conditions:

- Infinite hierarchy levels possible
- Reuse anything up to complete scenarios

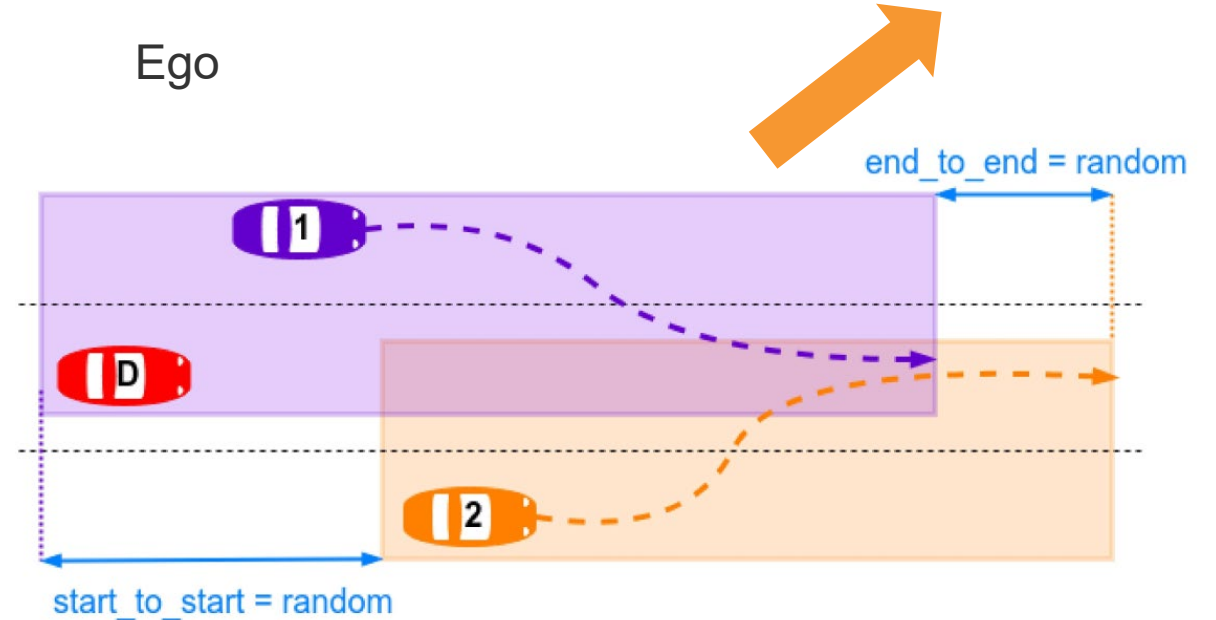
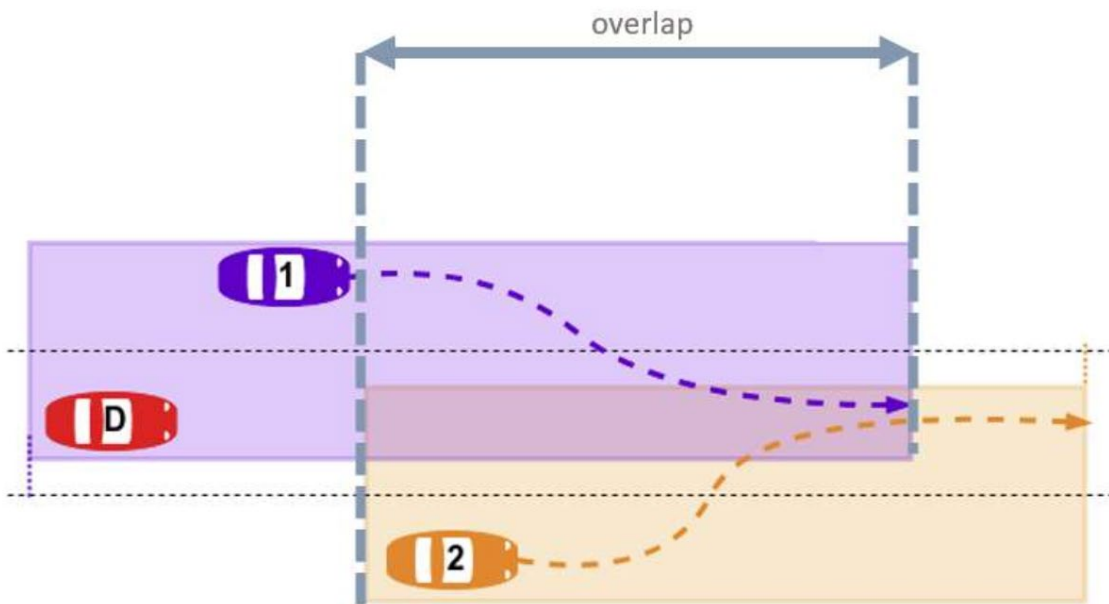


Scenarios are building blocks for more complex scenarios and tests.
Usage: ALKS multi vehicle interactions can be built upon single ALKS scenarios.

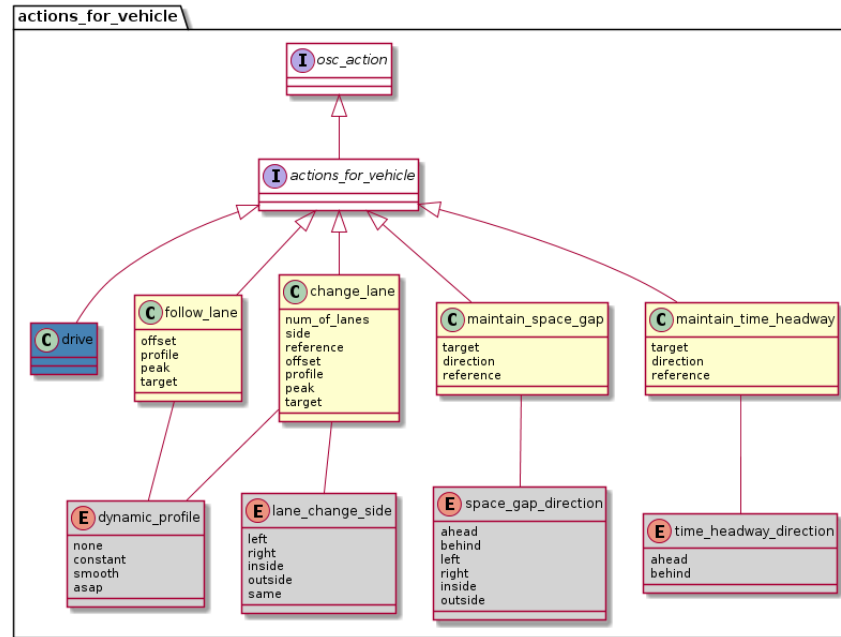
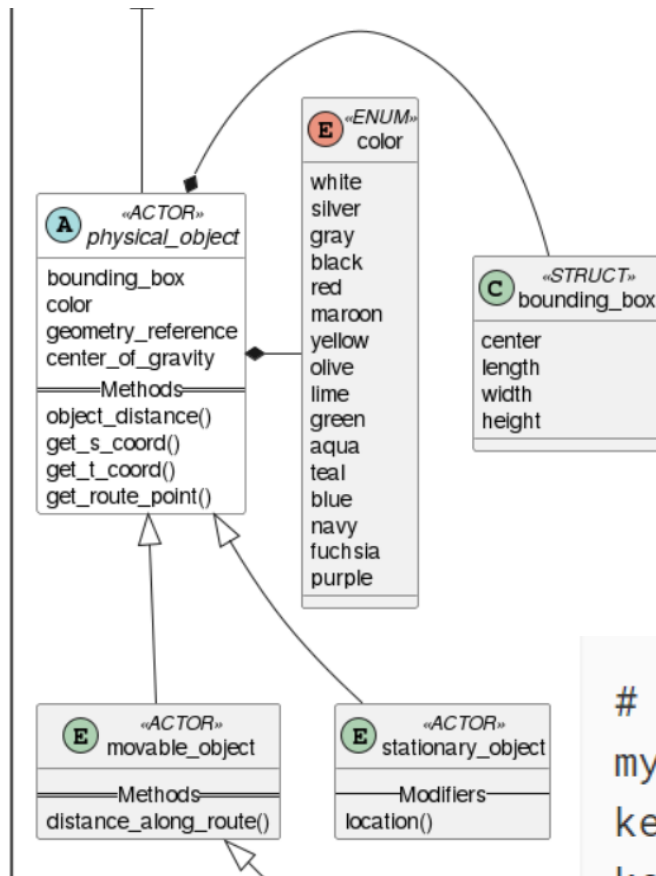
Various Mixing Options— Using “Parallel” operator



Ego



ASAM OpenSCENARIO 2.0 – Examples (Domain Model)



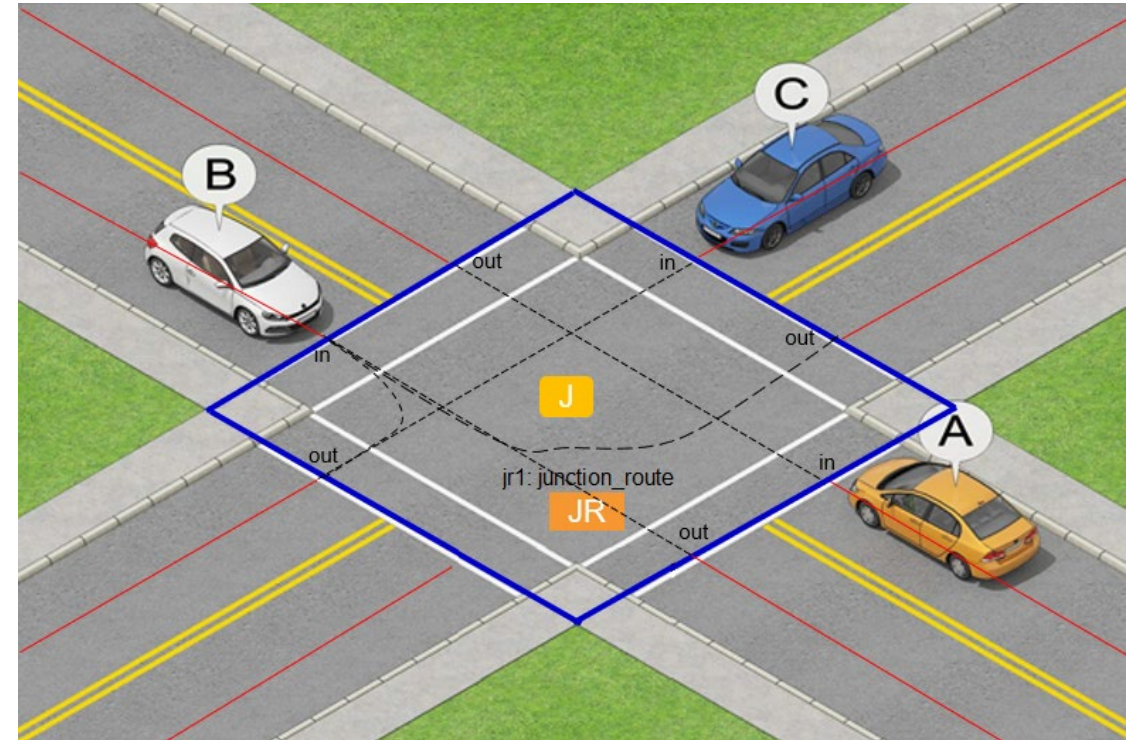
Key Difference:
 OSC 2.0 has a domain model which is extendible.
 OSC 1.x – has a data model. (defines actions, triggers)

```

# Constrain vehicle to be a motorcycle
my_motorcycle: vehicle
keep(my_motorcycle.vehicle_category == vru_vehicle)
keep(my_motorcycle.axles.size() == 2)
keep(my_motorcycle.axles[0].number_of_wheels == 1)
keep(my_motorcycle.axles[1].number_of_wheels == 1)
keep(my_motorcycle.intended_infrastructure[0] == driving)
    
```

ASAM OpenSCENARIO 2.0 – Road Abstraction

- An abstract description of the features of the road network that influence the behavior of the actors during the scenario. (mini-map)
- **Not a replacement for openDRIVE/real map.**
- **Creates a search space to find suitable location in a map.**
- **Usage: Scenario can automatically be placed on any location within the map, where these features exist.**
- Usage: Ability to describe actor behaviors on these features.



my_junction: junction

road_1, road_2, road_3, road_4: road

jr_12, jr_34: road

r1: map.roads_follow_in_junction(junction: my_junction, in_road: road_1, out_road: road_2, junction_route: jr_12)

r3: map.roads_follow_in_junction(junction: my_junction, in_road: road_3, out_road: road_4, junction_route: jr_34)

Built in risk mitigation:

- The standard is defining the language and domain model, while at the same time leaving room for implementation specific items.
- The standard offers significant extendibility capabilities, that can be used to bridge gaps.
- Examples:
 - Domain Model Extensions can be used to add entities and functionality.
 - External Functions calls and binding –can be used for various purposes – e.g. distribution functions.

ASAM OpenSCENARIO 2.0 - Out of Scope

System under test

The exact description of the system under test, e.g. detailed vehicle configuration, sensor placement, sensor models etc. is not part of OpenSCENARIO.

Behavioral model

The standard includes interface to behavioral models for actors. Those models are implementation specific.

Vehicle dynamics

Although the standard describes maneuvers in a kinematic way. The standard does not include all necessary elements to specify advanced motion dynamics.

Environmental models

The standard incorporates a domain actor to specify environment information and actions (e.g. rain, wind and more) but does not describe how this is to be interpreted by the testing platform.

Outline

- ASAM roadmap
- Difference between the two major versions (1.x, 2.0.0)
- Migration path – key points,

MIGRATION OSC 1.x to 2.0.0 - Key Points

- Migration guide is non-normative
- Guide covers OpenSCENARIO 1.0 with some exceptions.

- OpenSCENARIO 1.1 – parameter expressions and constraints covered automatically
 - Parameter distributions covered in guide
- Catalogs – handled

- Storyboard, event priorities

- Controllers – proposed model for migration to cover OpenSCENARIO 1.0 controller concept, up to simulation environment to implement in this way.
- Routes, trajectories – functionality is there, some naming changes, slight semantic differences (duration of actions).

MIGRATION OSC 1.x to 2.0.0 – Not covered in this release, can be implemented with coding or domain model extensions:

- AddEntity/RemoveEntity – incompatible concepts. Scenarios with intended purpose for physically possible situation can be coded with regular OpenSCENARIO 2.0 concepts. (i.e. manual conversion)
- TrafficSource/Sink – traffic generation works differently. Similar concepts can be coded to implement original intent of traffic swarm creation. (i.e. manual conversion)
- SelectTriggeringEntities – cannot be translated as simply as a flag in OpenSCENARIO 1.x ManeuverGroup. However, recreating the same result is possible when the scenario is coded with this consideration from start. (i.e. manual conversion)
- OpenSCENARIO 1.1 RoutePosition, TrajectoryPosition – Coding using road abstraction.
- TrafficSignal/TrafficSignalController pairs of actions and conditions – road network abstraction is a new concept in OpenSCENARIO 2.0.0 and the signalization part was not yet developed. Equivalent features are planned for next releases of the standard (manual non normative extension is possible now).
- Conditions: EndOfRoad, OffRoad, StandStill, TravelledDistance ((manual non normative extension is possible now , Standstill may be questionable).

MIGRATION OSC 1.x to 2.0.0 - Key Points

- SimulationTime – handling absolute time references is non-normative in the standard today, as any scenario can be used as a sub-scenario of another one. If problem can be reformulated with relative time references, ‘wait elapsed’ can be used.
 - If absolute time is necessary – domain model extension can be used through a simulation environment specific extension (top level clock)
- For now, two minor releases of the OpenSCENARIO 2 are in the plan:
 - OpenSCENARIO 2.0.1 – no major new development, should consist of small fixes and limited additions.
 - OpenSCENARIO 2.1 – a lot of features were in advanced development stage but were not ready for inclusion in 2.0 version of the standard. Depending on maturity, interest in a feature and project schedule, this version should bring more interesting additions and solutions for some known issues in the current release.

Proposal - MIGRATION OSC 1.x to 2.0.0 Q&A Forum

- As can be seen, some migrations topics have different implementation options.
- During implementers' forum – ASAM maintained a very successful Q&A forum for scenario coding.
- Proposal: ASAM to maintain (on github platform ?) a Q&A open platform to discuss and share OSC 1.x to OSC 2.0.0 migration knowledge and proposals.

Backup slides – TSC approval for project launch (circa 2020)

Technical Content - Features and Requirement from Concept Project

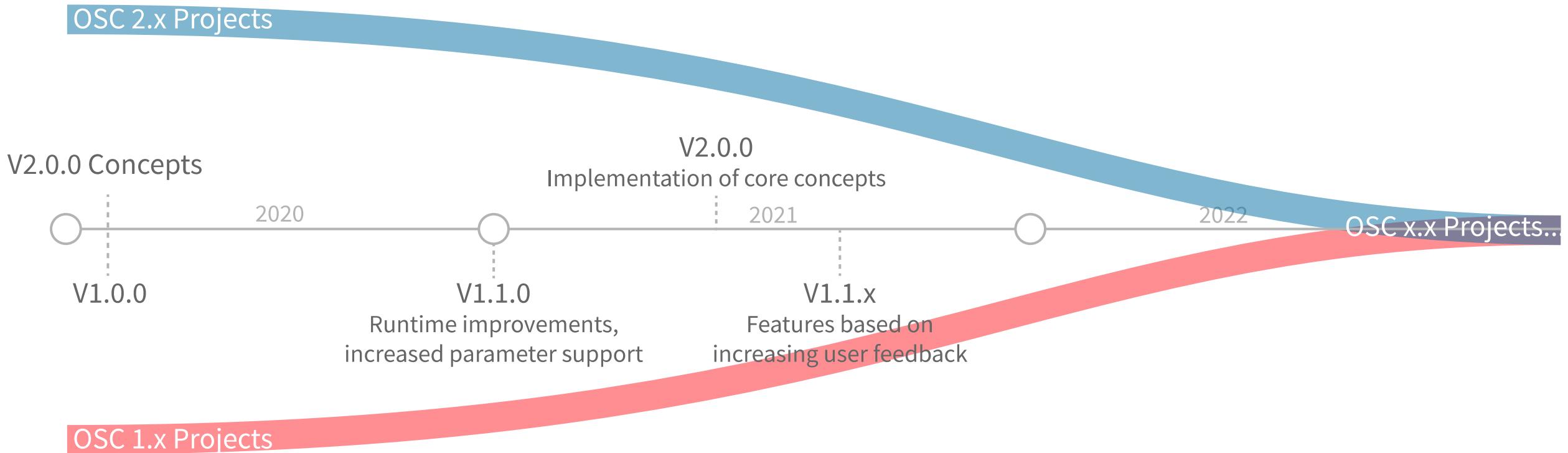
- OpenSCENARIO 2.0 needs to support:
 - Definition of tests and scenarios for the full development process of autonomous vehicles, and
 - the full complexity of real-world scenarios, including complex inner-city traffic.
- Requirements were redefined in the concept project as user stories from the perspectives of possible OpenSCENARIO users.
- This included:
 - Test engineers, tool developers, system engineers, safety consultants, government agencies, V&V development engineers and many more
 - See Proposal for full detail

Feature	Type
F001: Maneuver model	Change
F008: High level maneuver descriptions	New
F003: Traffic Model	New
F007: Parameter stochastics	New
F002: Driver Model	New
F004: Environmental Condition Model	New
F009: Replay of Recorded Scenarios	New
F010: Automatic parameter calculation	New
F005: Infrastructure Event Model	New
F006: Vehicle dynamics model	Change
F011: Additional metadata for parameters	New
F012: Language Constructs for Localization	New

Special Requirement: OSC 2.0/OSC 1.0

- OpenSCENARIO 2.0 is intended to be a full superset of the features of OpenSCENARIO 1.0. This means that a migration path for OpenSCENARIO 1.x scenarios to OpenSCENARIO 2.0 will be included in the release version of OpenSCENARIO 2.0.
- Migration of scenarios might require conversion to 2.0 syntax and semantics. **The run time behavior of any scenario converted from the latest OpenSCENARIO 1.x to OpenSCENARIO 2.0 shall be the same.**
- A conversion of OpenSCENARIO 2.0 scenarios to OpenSCENARIO 1.x will in general not be possible, since OpenSCENARIO 2.0 is intended to be a true superset of OpenSCENARIO 1.x. However, conversion of a subset of OpenSCENARIO 2.0 that maps to the feature set of OpenSCENARIO 1.x will be possible.
- The standard development projects for the 1.x and the 2.0 tracks shall ensure an up-to-date ruleset for a migration path from 1.x to 2.0.

Roadmap OpenSCENARIO

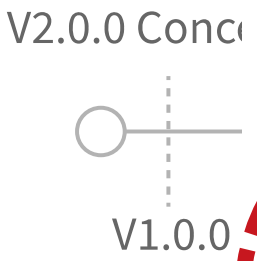


A convergence of the two versions will require further releases of OpenSCENARIO, which will be developed in subsequent OpenSCENARIO 1.x projects.

My analogy of the roadmap from USB 1.0 to type-C

OSC 2.

	USB 1.0 12mbps	USB 2.0 480mbps	USB 3.2 Gen 1 (Previously 3.0, then 3.1 Gen 1)	USB 3.2 Gen 2 (Previously 3.1 Gen 2)	USB 3.2 Gen 2x2 (Previously 3.2)
	12mbps	480mbps	5gbps	10gbps	20gbps
	 Type A Type B	 Type A Type B Mini-A	 Type A Type B	 Type A Type B	 Type-C



OSC x.x Projects..

OSC 1.

