OpenSCENARIO 2.0 Implementers Forum

Introduction to Domain Model activity in 2022-Q1

Nicolas Ochoa Lleras
DENSO Automotive

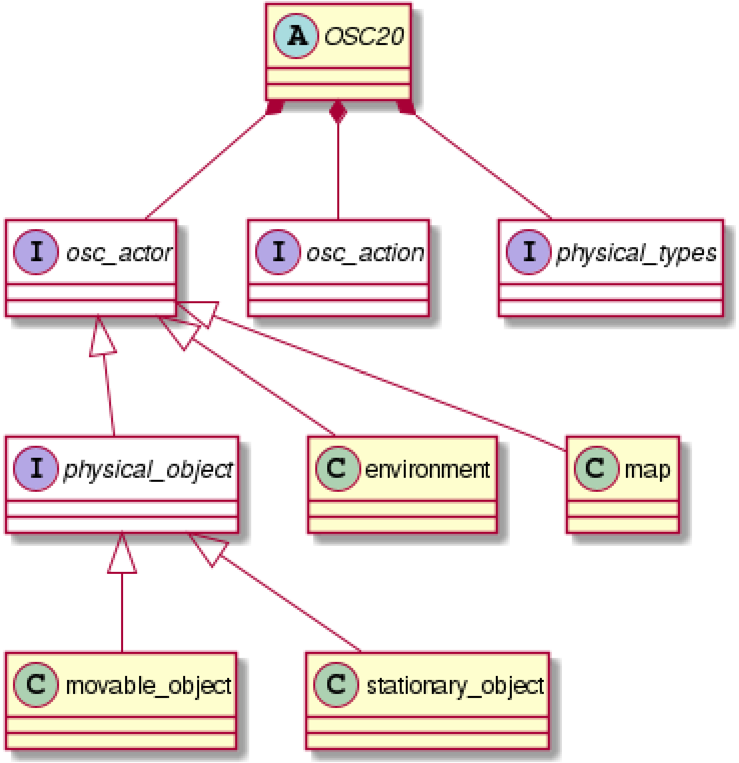
18. Januar 2022

ASAM Association for Standardization of Automation and Measuring Systems

The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

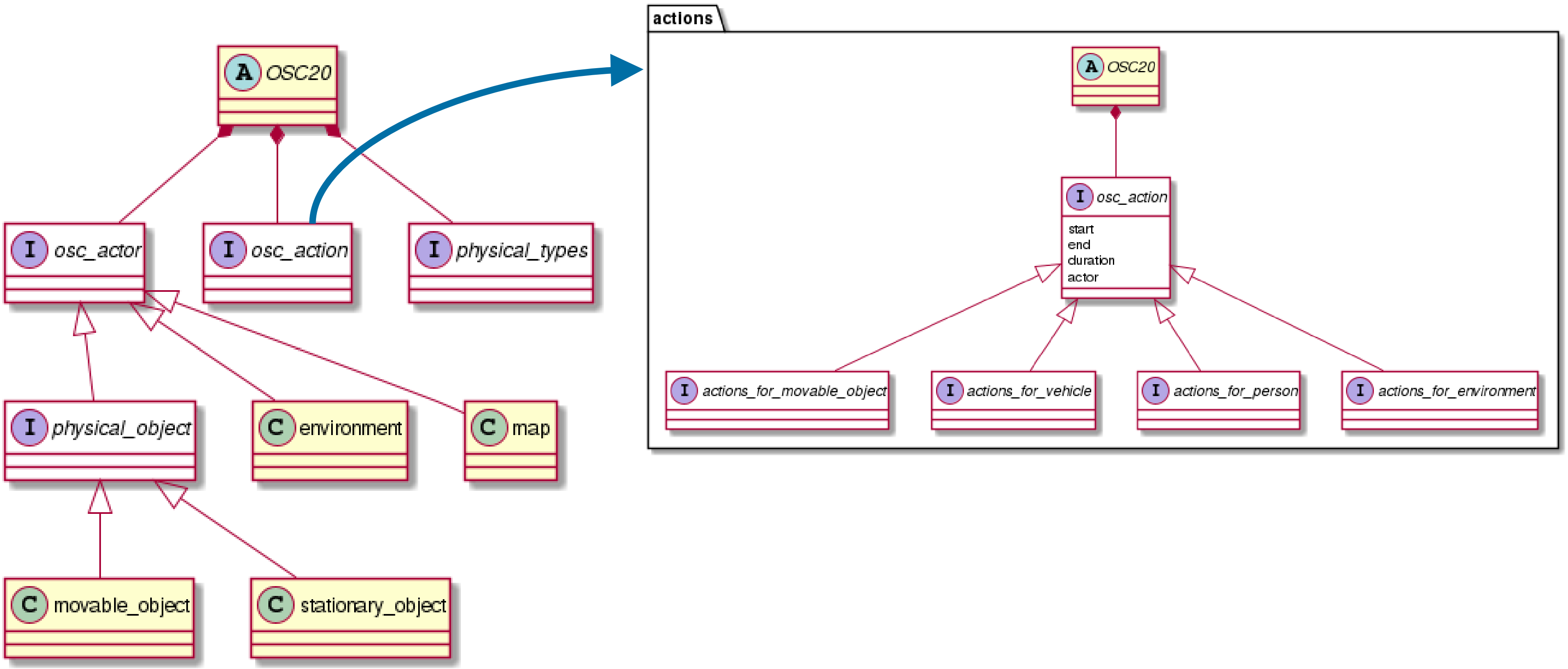
The Domain Model is user-extendable



The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

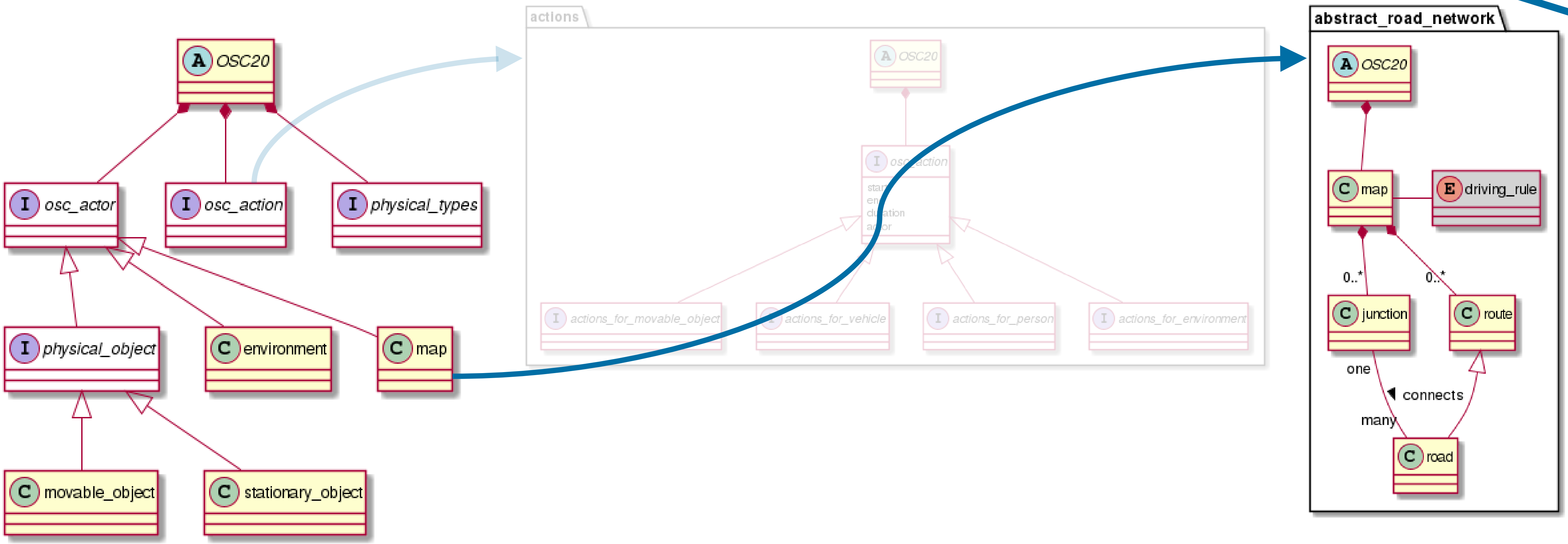
The Domain Model is user-extendable



The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

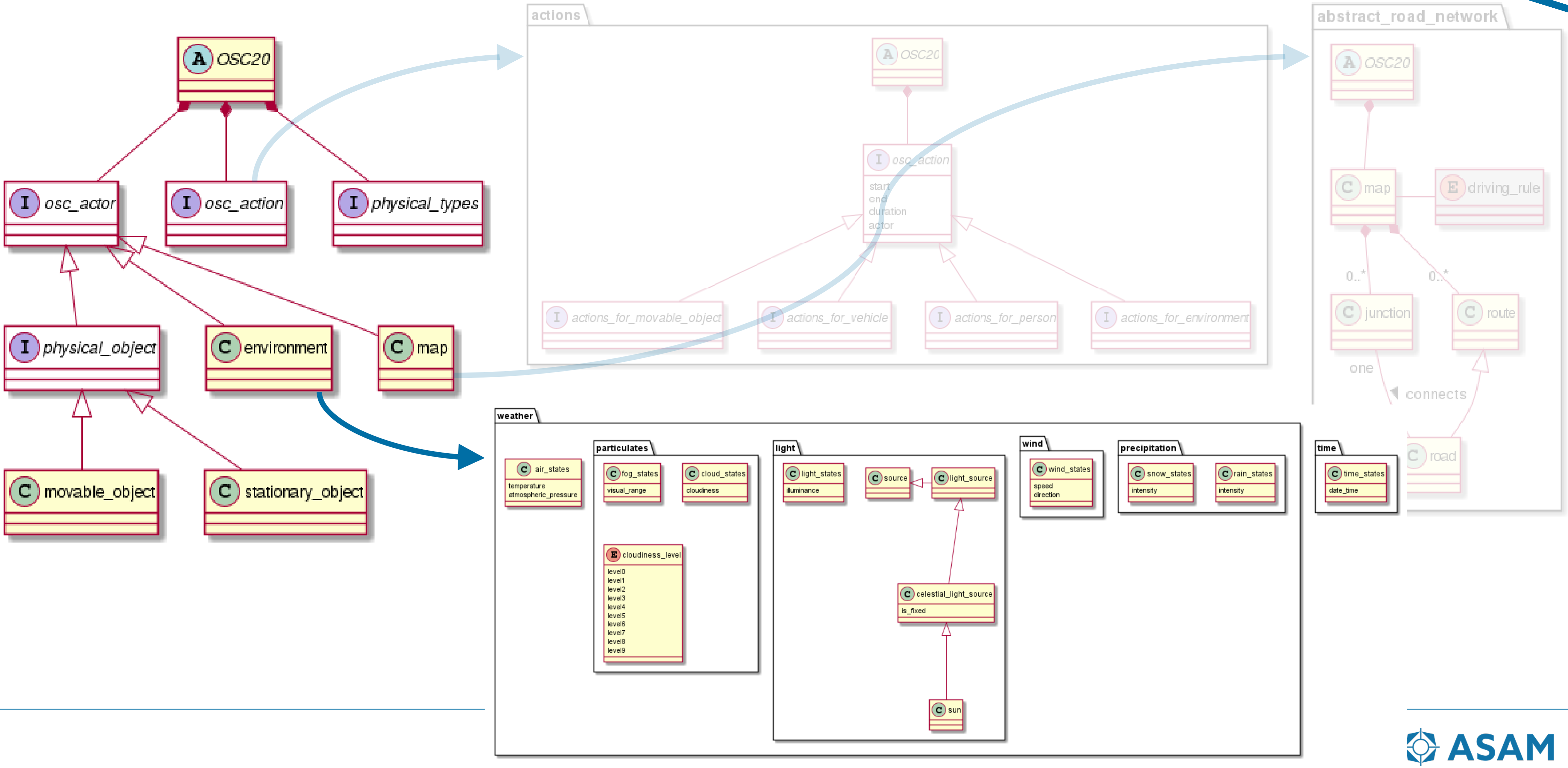
The Domain Model is user-extendable



The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

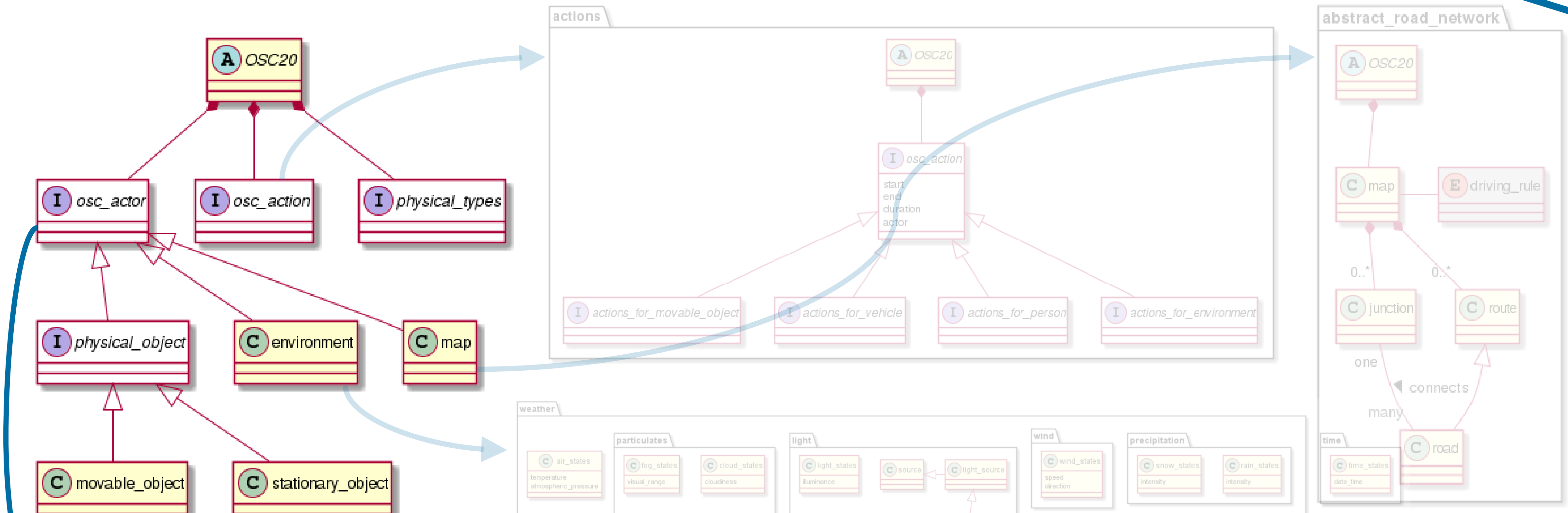
The Domain Model is user-extendable



The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

The Domain Model is user-extendable



Behavioral models struct

```
struct behavioral_model:
    bm_engine: bm_engine
```

Motion controller example

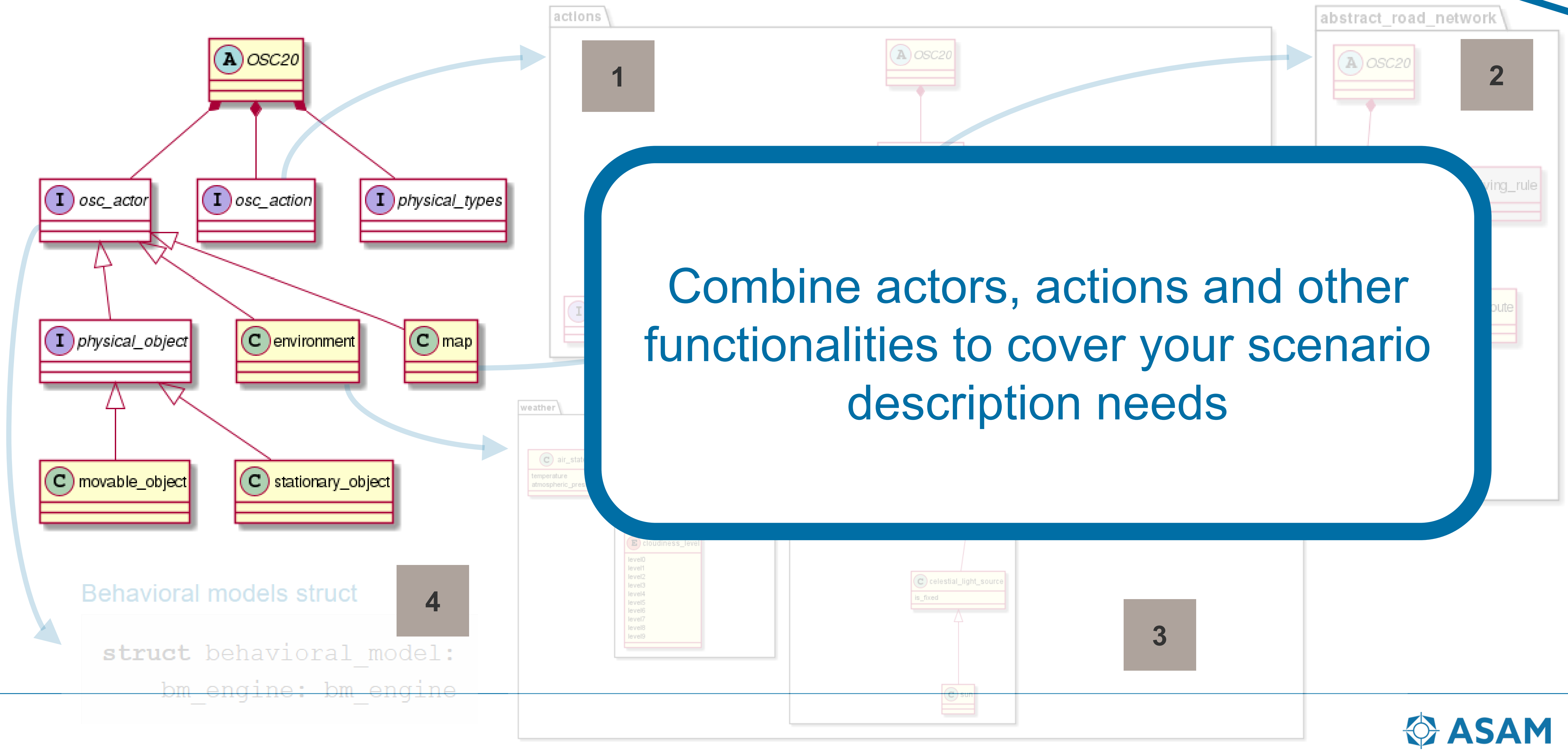
```
motion_control inherits behavioral_model:
    longitudinal: bool
    lateral: bool

my_controller: motion_control with:
    keep(longitudinal == True)
    keep(lateral == True)
```

The Domain Model is the “Standard Library” of OSC 2.0

Applies the OSC 2.0 language features to offer the fundamental functionality for scenario description

The Domain Model is user-extendable



Combine actors, actions and other functionalities to cover your scenario description needs

Behavioral models struct

```
struct behavioral_model:
    bm_engine: bm_engine
```

During 2022-Q1 we must evaluate the usability of the “library”

Include any critical improvements before the final OSC 2.0 release

	Can users employ the OSC 2.0 DM to:	Key questions 2022-Q1
1	Describe well-known scenarios, like LKAS, NCAP, etc.	Combining **actions**
2	Create abstract descriptions of commonly-used road networks	New concept for **routes**
3	Control environmental conditions	**Environment** as an actor
4	Interact with implementation modules through the Behavioral Models	Expansion of **controllers**

Invitation to participate in the
OSC 2.0 Implementer’s Forum

Clarity of the
documentation

Desired outcomes to improve OSC 2.0 spec before final release

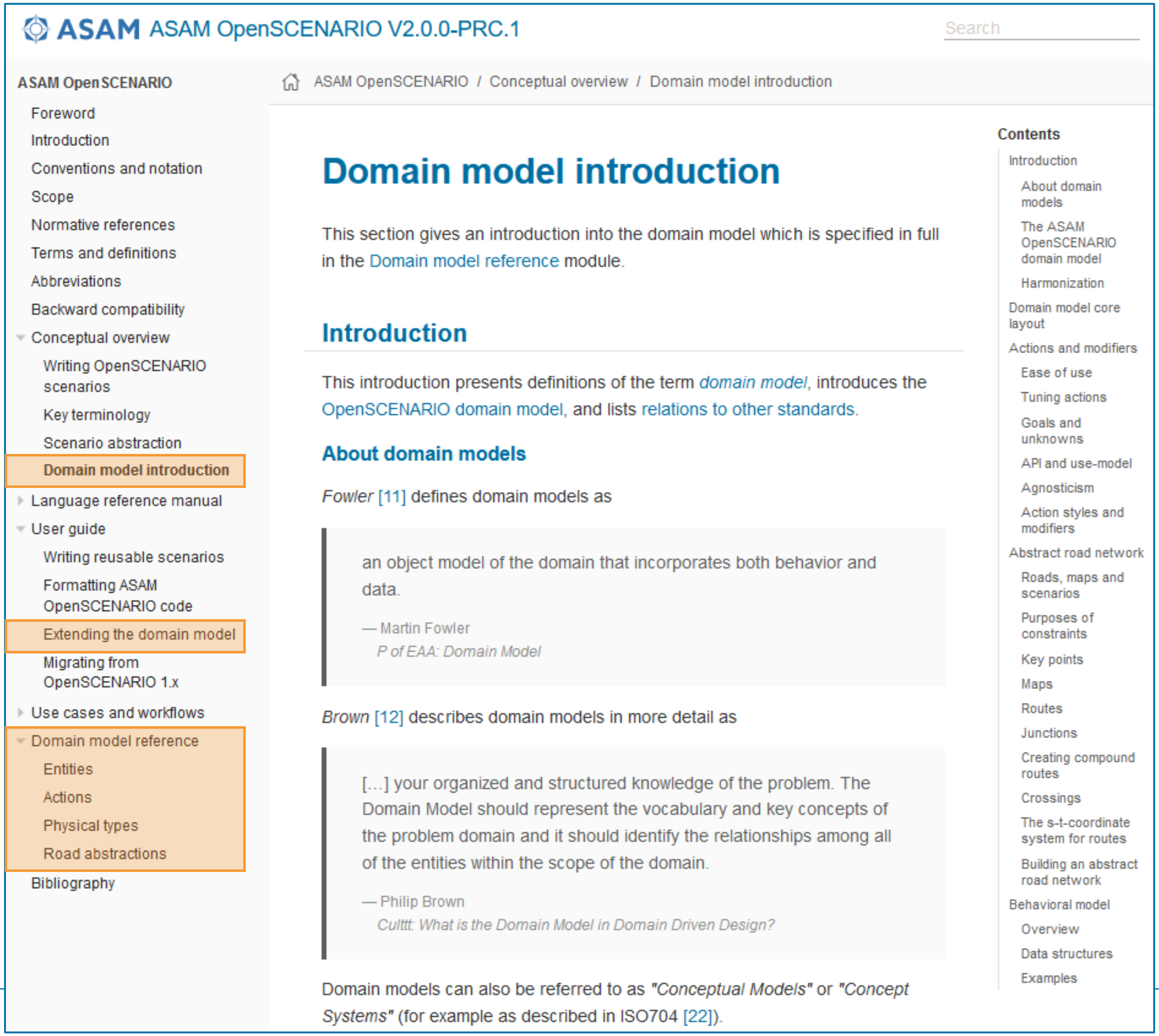
	Collaboration: Implementer’s Forum (IF) and Domain Model (DM)	
1	**Fix issues identified in review period (November 2021)**	DM responsibility
2	**Improve the examples that are delivered with the specification**	IF + DM sessions
3	**Critical improvements to documentation**	DM to implement feedback from IF
4	**Fix critical inconsistencies** • Within the domain model • Between the domain model and language features	DM to implement feedback from IF

This phase will **NOT** expand functionality of OSC 2.0

The intention is to improve the **clarity** and **consistency** of the OSC 2.0 specification

Invitation to participate in the **OSC 2.0 Implementer’s Forum**

Implementers Forum working mode: OSC 2.0 spec + scenario => osc2 file



ASAM ASAM OpenSCENARIO V2.0.0-PRC.1

Search

ASAM OpenSCENARIO
- Foreword
- Introduction
- Conventions and notation
- Scope
- Normative references
- Terms and definitions
- Abbreviations
- Backward compatibility
- Conceptual overview
  - Writing OpenSCENARIO scenarios
  - Key terminology
  - Scenario abstraction
  - Domain model introduction
- Language reference manual
- User guide
  - Writing reusable scenarios
  - Formatting ASAM OpenSCENARIO code
  - Extending the domain model
  - Migrating from OpenSCENARIO 1.x
- Use cases and workflows
- Domain model reference
  - Entities
  - Actions
  - Physical types
  - Road abstractions
- Bibliography

ASAM OpenSCENARIO / Conceptual overview / Domain model introduction

Domain model introduction

This section gives an introduction into the domain model which is specified in full in the Domain model reference module.

Introduction

This introduction presents definitions of the term *domain model*, introduces the OpenSCENARIO domain model, and lists relations to other standards.

About domain models

Fowler [11] defines domain models as

> an object model of the domain that incorporates both behavior and data.
>
> — Martin Fowler
> *P of EAA: Domain Model*

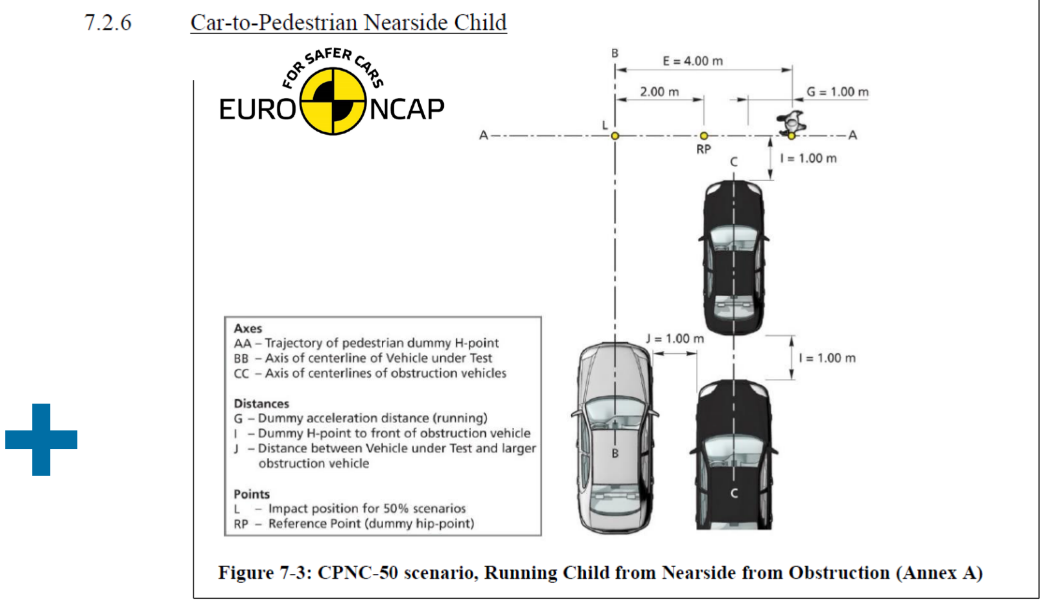
Brown [12] describes domain models in more detail as

> [...] your organized and structured knowledge of the problem. The Domain Model should represent the vocabulary and key concepts of the problem domain and it should identify the relationships among all of the entities within the scope of the domain.
>
> — Philip Brown
> *Culttt: What is the Domain Model in Domain Driven Design?*

Domain models can also be referred to as *"Conceptual Models"* or *"Concept Systems"* (for example as described in ISO704 [22]).

Contents

+



7.2.6 Car-to-Pedestrian Nearside Child

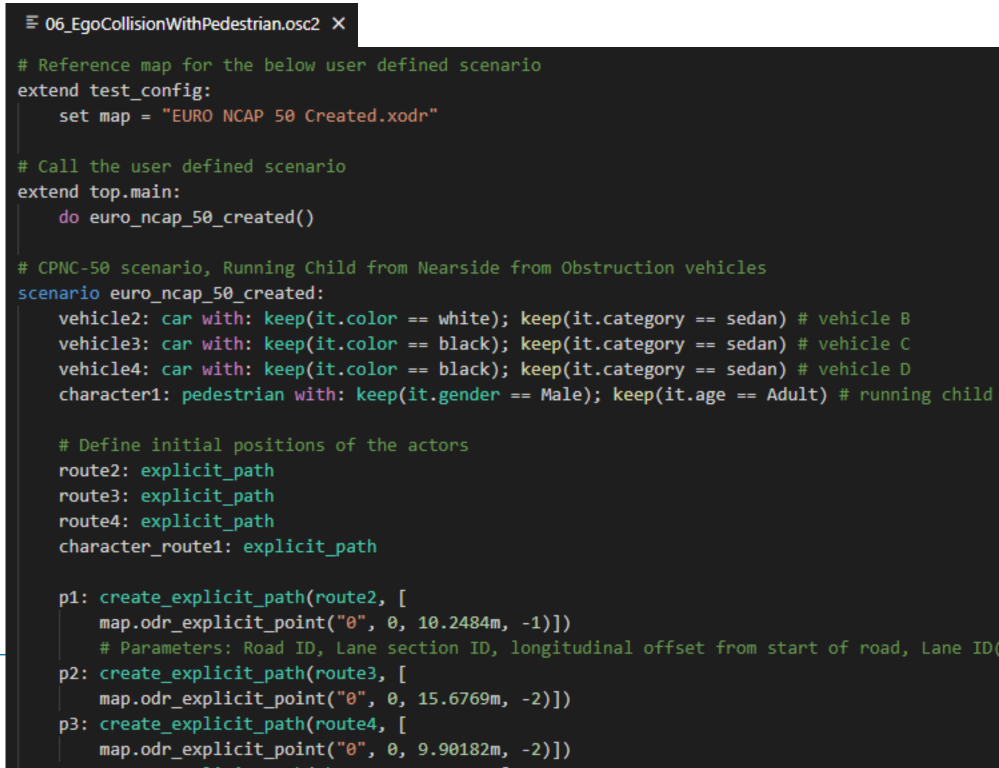
Axes
AA – Trajectory of pedestrian dummy H-point
BB – Axis of centerline of Vehicle under Test
CC – Axis of centerlines of obstruction vehicles

Distances
G – Dummy acceleration distance (running)
I – Dummy H-point to front of obstruction vehicle
J – Distance between Vehicle under Test and larger obstruction vehicle

Points
L – Impact position for 50% scenarios
RP – Reference Point (dummy hip-point)

Figure 7-3: CPNC-50 scenario, Running Child from Nearside from Obstruction (Annex A)

=>



06_EgoCollisionWithPedestrian.osc2 ✕

```
# Reference map for the below user defined scenario
extend test_config:
    set map = "EURO NCAP 50 Created.xodr"

# Call the user defined scenario
extend top.main:
    do euro_ncap_50_created()

# CPNC-50 scenario, Running Child from Nearside from Obstruction vehicles
scenario euro_ncap_50_created:
    vehicle2: car with: keep(it.color == white); keep(it.category == sedan) # vehicle B
    vehicle3: car with: keep(it.color == black); keep(it.category == sedan) # vehicle C
    vehicle4: car with: keep(it.color == black); keep(it.category == sedan) # vehicle D
    character1: pedestrian with: keep(it.gender == Male); keep(it.age == Adult) # running child

    # Define initial positions of the actors
    route2: explicit_path
    route3: explicit_path
    route4: explicit_path
    character_route1: explicit_path

    p1: create_explicit_path(route2, [
        map.odr_explicit_point("0", 0, 10.2484m, -1)])
        # Parameters: Road ID, Lane section ID, longitudinal offset from start of road, Lane ID(
    p2: create_explicit_path(route3, [
        map.odr_explicit_point("0", 0, 15.6769m, -2)])
    p3: create_explicit_path(route4, [
        map.odr_explicit_point("0", 0, 9.90182m, -2)])
```

Benjamin Engel
Global Technology Manager, ASAM e.V.

Phone: +49 1516 1645 936
Email: benjamin.engel@asam.net