Continental
The Future in Motion

Open Scenario Code Extensions

How to deal with complex Scenarios

SensePlanAct

About me

- 7 years of Emergency Brake Assist Development
- Part-Time Evaluation Toolchain Development
- Now Full-Time Team-Leader Test Tooling for Highway-Pilot

- OSC Experience
  - Integrated Open Drive Support into own Tool
  - Integrated Open Scenario Support into own Tool

Questions

Please raise Hands

- Who has ever created his own Open Scenario File from Scratch?
- Who has ever written Code that interacted with Open Scenario?

Open Scenario

Goal

Open Scenario should allow the Exchange of Dynamic Traffic Object Behaviour.

It should be **UNAMBIGUOUS.**

It should be **EASY TO LEARN.**

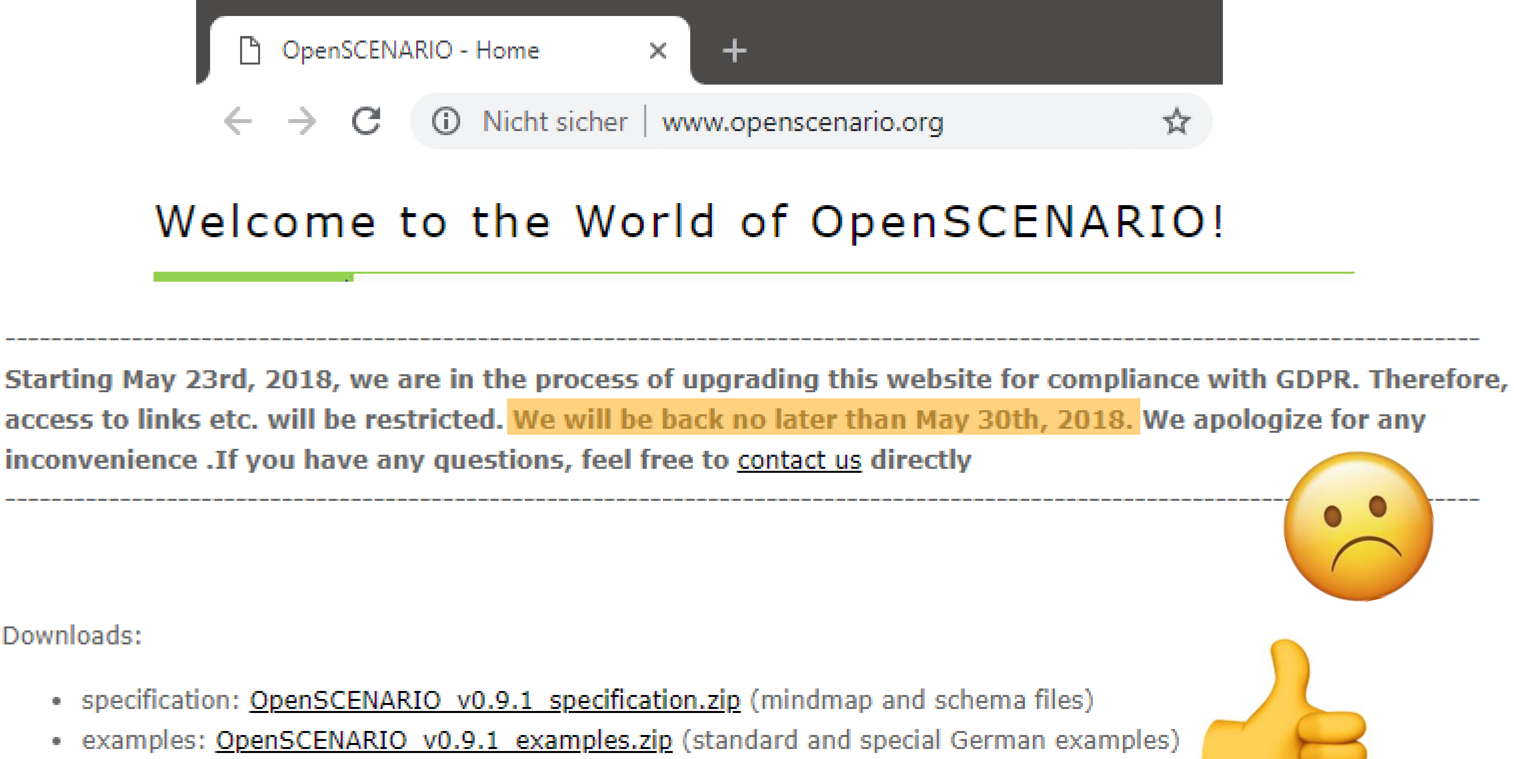
It should be **EASY TO INTERPRET.**

It should be **WELL DOCUMENTED.**

It should be **FLEXIBLE.**

It should be **STABLE.**

Open Scenario Experience

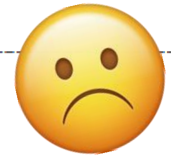


Welcome to the World of OpenSCENARIO!

Starting May 23rd, 2018, we are in the process of upgrading this website for compliance with GDPR. Therefore, access to links etc. will be restricted. We will be back no later than May 30th, 2018. We apologize for any inconvenience .If you have any questions, feel free to contact us directly

Downloads:

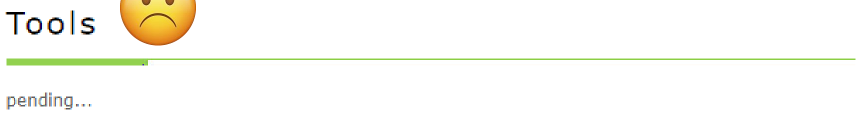
- specification: OpenSCENARIO_v0.9.1_specification.zip (mindmap and schema files)
- examples: OpenSCENARIO_v0.9.1_examples.zip (standard and special German examples)



Open Scenario

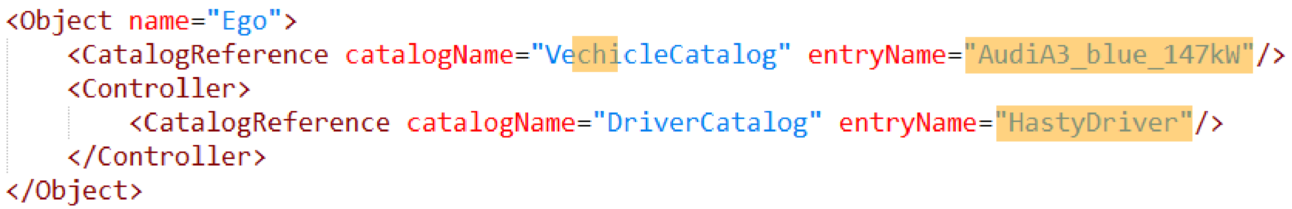
Experience

- Specification
  - Xsd Files 👍
  - No Documentation ☹️
- Examples
  - Corrupt ☹️


Tools ☹️

pending...

OpenSCENARIO_v0.9.1_examples\Standard\LaneChanger\LaneChanger.xosc



```
<Object name="Ego">
    <CatalogReference catalogName="VechicleCatalog" entryName="AudiA3_blue_147kW"/>
    <Controller>
        <CatalogReference catalogName="DriverCatalog" entryName="HastyDriver"/>
    </Controller>
</Object>
```

Open Scenario

Shortcomings

- **Support**
  - Website offline since May 2018
  - Official Examples are corrupt
  - Lack of Tool-Support
  - No Documentation about the Format
  - No Reference-Implementation
- **Format**
  - Ambiguous
    - Models have to be exchanged as well
    - Same Trigger different Behaviour
    - Vehicle Model vs. Trajectory
  - Difficult Integration
  - Complex even for simple Interactions
  - Scenarios of medium/high Complexity (i.e. cut-off with controlling Behaviour) impossible to describe

Open Scenario should allow the Exchange of Dynamic Traffic Object Behaviour.

It should be **UNAMBIGUOUS.**
It should be **EASY TO LEARN.**
It should be **EASY TO INTERPRET.**
It should be **WELL DOCUMENTED.**
It should be **FLEXIBLE.**
It should be **STABLE.**

What works in the static world of OpenDrive does not translate into the dynamic, complex, interactive world of Open Scenario

Open Scenario **Code Extensions**

OSC Code Extensions

Definition in OSC

- New Section in XOSC File
- Backward compatible
- Languages (Proposal)
  - Python
  - C/C++ via shared Libraries

```xml
<?xml version="1.0" encoding="utf-8"?>
<OpenSCENARIO>
  <FileHeader>
  <!--Parameter Section-->
  <ParameterDeclaration>
  <!--Catalogs Section-->
  <Catalogs>
  <!--Roadnetwork Section-->
  <RoadNetwork>
  <!--Entities Section-->
  <Entities>
  <!--CodeEntities Section-->
  <CodeEntities>
      <CodeEntity name="Ego" type="python3" filepath="./driverSimple.py">
          <ParameterDeclaration>
              <Parameter name="InitPosX"        type="double" value="20.0"/>
              <Parameter name="InitPosY"        type="double" value="6.25"/>
              <Parameter name="InitOrientation" type="double" value="0.0"/>
              <Parameter name="InitVelocity"    type="double" value="4.0"/>
          </ParameterDeclaration>
      </CodeEntity>
      <CodeEntity name="RightLaneDriver" type="cdll" filepath="../../bin/driverSimple.dll">
          <ParameterDeclaration>
              <Parameter name="InitPosX"        type="double" value="30.0"/>
              <Parameter name="InitPosY"        type="double" value="2.75"/>
              <Parameter name="InitVelocity"    type="double" value="3.0"/>
              <Parameter name="Width"           type="double" value="1.8"/>
              <Parameter name="Length"          type="double" value="4.8"/>
          </ParameterDeclaration>
      </CodeEntity>
  </CodeEntities>
  <!--Storyboard Section-->
  <Storyboard>
</OpenSCENARIO>
```

Example

Open Scenario Code Extensions

Interface Proposal

Entity Information
- Type
- Dimensions

Initial Kinematic
- Position / Orientation
- Velocity / Acceleration

Custom Data

Called once before Simulation

Static Environment Data
- Lanes
- Objects

Parameters
- Defined in *.xosc

Custom Data

```
InitOutput    = Init   (InitInput)
ProcessOutput = Process(ProcessInput)
```

Updated Kinematic
- Position / Orientation
- Velocity / Acceleration

Custom Data

Called every step of Simulation

Simulation Data
- Time
- Time Delta
- Cycle Number

Dynamic Objects
- Classic Entities
- Code Entities

Custom Data

Open Scenario Code Extensions

Example

```python
import math

class VehicleModelSimple():
    def __init__(self):
        self.posX         = 0.0
        self.posY         = 0.0
        self.orientation  = 0.0
        self.velocity     = 0.0
        self.acceleration = 0.0

    def Step(self, timeStep):
        # Update Velocity
        self.velocity += self.acceleration * timeStep
        # Calculate Distance
        distance = self.velocity * timeStep
        # Move Distance in Direction of Orientation
        self.posX += distance * math.cos(self.orientation)
        self.posY += distance * math.sin(self.orientation)
```

Simple Vehicle Model

Simple Follower

```python
import vehModelSimple
class SimpleFollower():
    def __init__(self):
        self.model = vehModelSimple.VehicleModelSimple()
        # Initialize underlying Vehicle-Model
        self.model.posX         = 20.0
        self.model.posY         = 2.75
        self.model.velocity     = 6.0
        self.model.acceleration = 0.0
        self.width              = 1.8
        self.length             = 4.8

    def Init(self, InitInput):
        return self.model

    def Process(self, ProcessInput):
        # Calcuate realtive longitudinal Distance to Ego Vehicle
        distToEgo = ProcessInput.dynObjects["Ego"].posY - self.model.posY
        # Follow Logic
        if distToEgo > 20.0:
            self.model.acceleration = 0.2
        else:
            self.model.acceleration = -0.2
        # Advance vehicle model
        self.model.Step(ProcessInput.simData.simTimeDelta)
        return self.model
```

Open Scenario Code Extensions

Advantages

- **UNAMBIGUOUS**
  Function Calls always return one well defined Result
- **EASY TO LEARN**
  Programming Skills already wide spread and especially Python easy to learn
- **EASY TO INTERPRET**
  Extending existing Solutions with Python or Shared Libraries is common Practice
- **WELL DOCUMENTED**
  OSI protobuf
- **FLEXIBLE**
  Simple Scenarios are simple. Unlimited complexity is possible without interface change
- **STABLE**
  Chance of minimal alignment Effort

- Wide Support of Tools for Editing
- Powerful Mechanisms like Calling functions, Inheritance, Library support, Debugging
- Protected Exchange via compiled libraries possible

Thank you!