TRIANGRAPHICS
Intelligent Terrain Solutions

ASAM CheckerLib

Mirco Nierenz

TrianGraphics GmbH, Managing Director

23. 01. 2024, Höhenkirchen

ASAM Quality Checker KickOff

Agenda

- Motivation
- Content
  - Categories of checks
  - Overview of current checks
  - File Structure
- Implementation
  - Data Structure, Classes
  - Process flow
- Live Demo
  - CheckerLib
  - QChecker Report
  - Simple Webviewer
- Next Steps
  - Found Issues, Todo’s
  - Questions

Motivation

- Many tool and data providers have their own test suites, but
  - checks from their point of view or application
  - Incomplete
  - Partly not accessible
- Data provider generate and Tools read formats differently
  - interpret elements slightly differently
  - do not support all features or characteristics
  - sometimes have certain constraints
- Standard description not always clear
  - offer room for interpretation
  - Implementation examples are missing
  - Schema does not cover everything or is incomplete

ASAM CheckerLib

Content of CheckerLib

Content

Categories

- Basis
  - File exists
  - File readable (as XML)
- Schema
  - Read Version
  - Test against schema file
- Semantic
  - Linkage
  - Order
  - Ranges
- Geometry
  - Values correct (e.g. lengths)
  - Values in range
  - Steadiness

Content

Categories

- **OpenMSL** - https://github.com/openMSL
  - extension of ASAM CheckerLib
  - Opensource
  - available after the end of the GaiaX research project
- Tool Compatibility
  - loadable / usable in applications
  - Special requirements of applications
- Linkage
  - to other OpenStandards
  - Correct references / position
- Statistic
  - Node elements (e.g. roads / junctions)
  - Objects (e.g. signal type)

Content

Checks OpenDRIVE

- Basic
  - check_file_exist
  - check_xml_parsing
- Schema
  - check_version
  - check_schema
  - check_ID_type
  - check_lane_ID_type
- Geometry
  - check_road_min_length
  - check_road_geometry_length
  - check_paramPoly3
- Semantic
  - check_road_linkage
  - check_road_linkage_backward
  - check_laneSection_min_length
  - check_laneSection_valid_s
  - check_lanes_order
  - check_lanes_id_linkage
  - check_lane_validity
  - check_lane_valid_width
  - check_lane_valid_sOffset
  - check_lane_type
  - check_junction_connections
  - check_junction_lane_linkage
  - check_junction_lane_linkage_order
  - check_driving_lanes_continue_in_junction
  - check_signal_position
  - check_signal_size
  - check_object_position
  - check_object_size

Content

Checks OpenSCENARIO

- Basic
  - check_file_exist
  - check_xml_parsing
- Schema
  - check_version
  - check_schema

Content

File Structure

- ASAM-CheckerLib - https://github.com/asam-ev/qc-pyFramework
  - main script files
  - **[Format extensions]** – like xodr, xosc
    - Config.json - configuration of each check, e.g. epsilon values, min/max ranges
    - Format.json – name, description, extension of format
    - **Doc**
      - Documentation of current checks
      - Tables for possible checks
    - **checks**
      - __init__.py - script to define order of categories
      - **[Categories]**
        - __init__.py script to define category/bundle name and order of checks
        - check_[name].py – check scripts
    - **Examples**
      - **[Categories]**
        - Examples files for testing for each check
  - **doc**
    - presentations

Implementation of CheckerLib

Implementation

General

- **Language**
  - Python
- **Execution**
  - Different platforms (Linux, Windows, Mac, Web?)
  - Local or
  - Backend (Server)
  - Docker Container
    - Tested in OVAL-Platform of Perpettum Progress GmbH
    - Tested in GitHub Pipeline Environment from asc(s

Implementation

Data Structure

- Each Format has
  - Bundles (categories) of
    - Checks with
      - configurations (e.g. epsilon)
      - and report
        - Issues
          - IssueLevel
          - Locations

- **IssueLevel**
  - ERROR, WARNING, INFORMATION
- **Location**
  - FileLocation (type, row, column)
  - XMLLocation (xpath)
  - RoadLocation(roadID, s, t)
  - can be combined
  - Help functions for simple creation
- **Issue**
  - ID,description, IssueLevel, list of locations
- **Checker**
  - ID, description, List of Issues
- **CheckerBundle**
  - name, description, version, list of Checker, configurations
- **ResultReport**
  - List of CheckerBundles, FilePath

Implementation

Call

- Call Parameters
  - INPUT_FILES
    - can be a mix of different formats
  - -a addition-check-dirs
    - Reference to additional Checker Bundles
  - -c config
    - Path to config file. Specification of external variables for checks
  - -t output-type
    - Output format of result report: xqar, json, txt
  - -o output-directory
    - Path to validation report folder

- example
  - python   main.py   xodr/examples/Schema/check_no_lanes.xodr

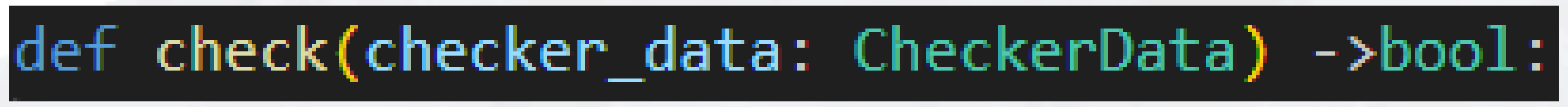
- Input
  - One or more files / folders
  - Currently XML based formats with schema
    - OpenDRIVE
    - OpenSCENARIO

- Output
  - Write as
    - **QChecker XQAR**
    - TXT
    - JSON
  - Console

Implementation

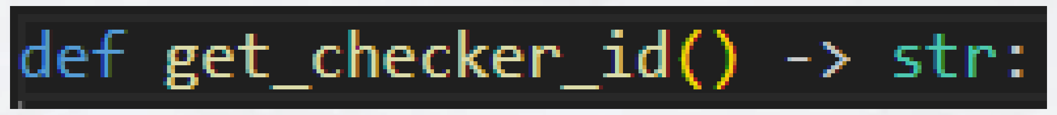
Classes

- Classes
  - Main.py
    - reads, checks parameters and for each file
      - calls validation and writes output
  - validator.py
    - loads registered bundles and handle order
    - loads and executes check
  - result_report.py
    - Data structure for report file and functions for issue registering
    - Different writes function for Report Tree
  - interface functions for Checks
    - check function



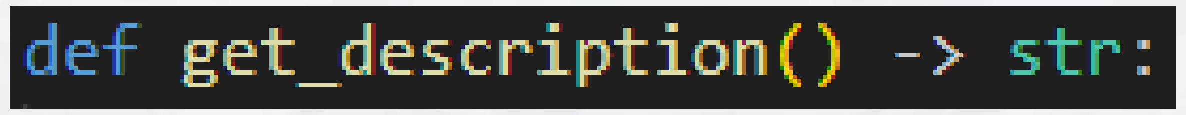
```
def check(checker_data: CheckerData) ->bool:
```

    - name of check



```
def get_checker_id() -> str:
```

    - description of check



```
def get_description() -> str:
```

Implementation

Process flow

- main.py
  - analysis of the parameters
  - For each input file
    - validate function in Validate.py
      - Get file extension and choose format folder
      - run_checks function
        - Load/register category scripts
        - For each category
          - Load/register check functions
          - For each check
            - Run check function in check_[name].py
            - Checks for abort
    - Write result
    - Checks for abort
  - Write issues counter

Implementation

Error handling

- Critical errors can cause an abort
  - E.g. file not exist or not readable
- Error in check_schema can also be aborted
  - configurable in config.json -> Schema -> check_Schema -> abort_if_error
- Cancellation of the entire script after an error
  - configurable via parameter –e -> choices=['no-exit', 'exit-if-error']

Implementation

Criteria's

- **Extendable**
  - Checks
    - add check_[name].py file to appropriate category of format
    - Implement necessary functions
    - If necessary, adjust the configuration and order
  - Category/Bundle
    - Add category folder
    - create __init__.py with description and order
    - Add checks
  - Format
    - Add format folder
    - Create fomat.json and config.json
    - Add Categories/Bundles

- **Completeness**
  - ➢ possible at all or a goal of this project?
  - ASAM Community with different experiences and points of view

- **Uniform**
  - standardized interface for checks for different formats

- **Simplicity**
  - Examples
    - One invalid data example per check
    - Only the specific issues of this check hits
    - Valid data examples
  - detailed documentation of each check
  - easily accessible using Python
  - Opensource

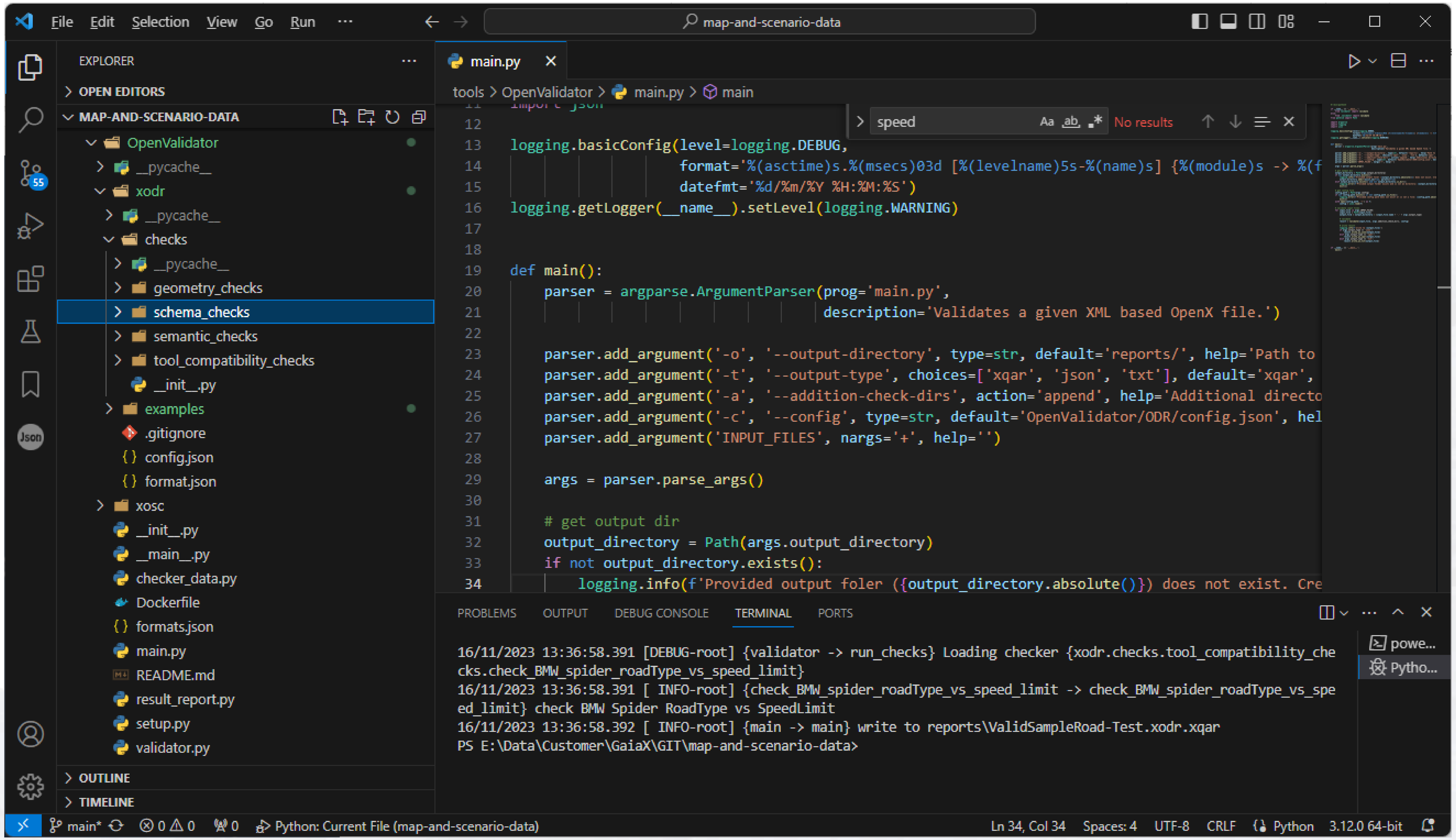
- **Flexible**
  - Support different formats (not only XML)
  - Configurable
  - Can be integrated into automated pipelines
  - One or more files or entire folders can be tested

ASAM CheckerLib

Live Demo

Live Demo

CheckerLib

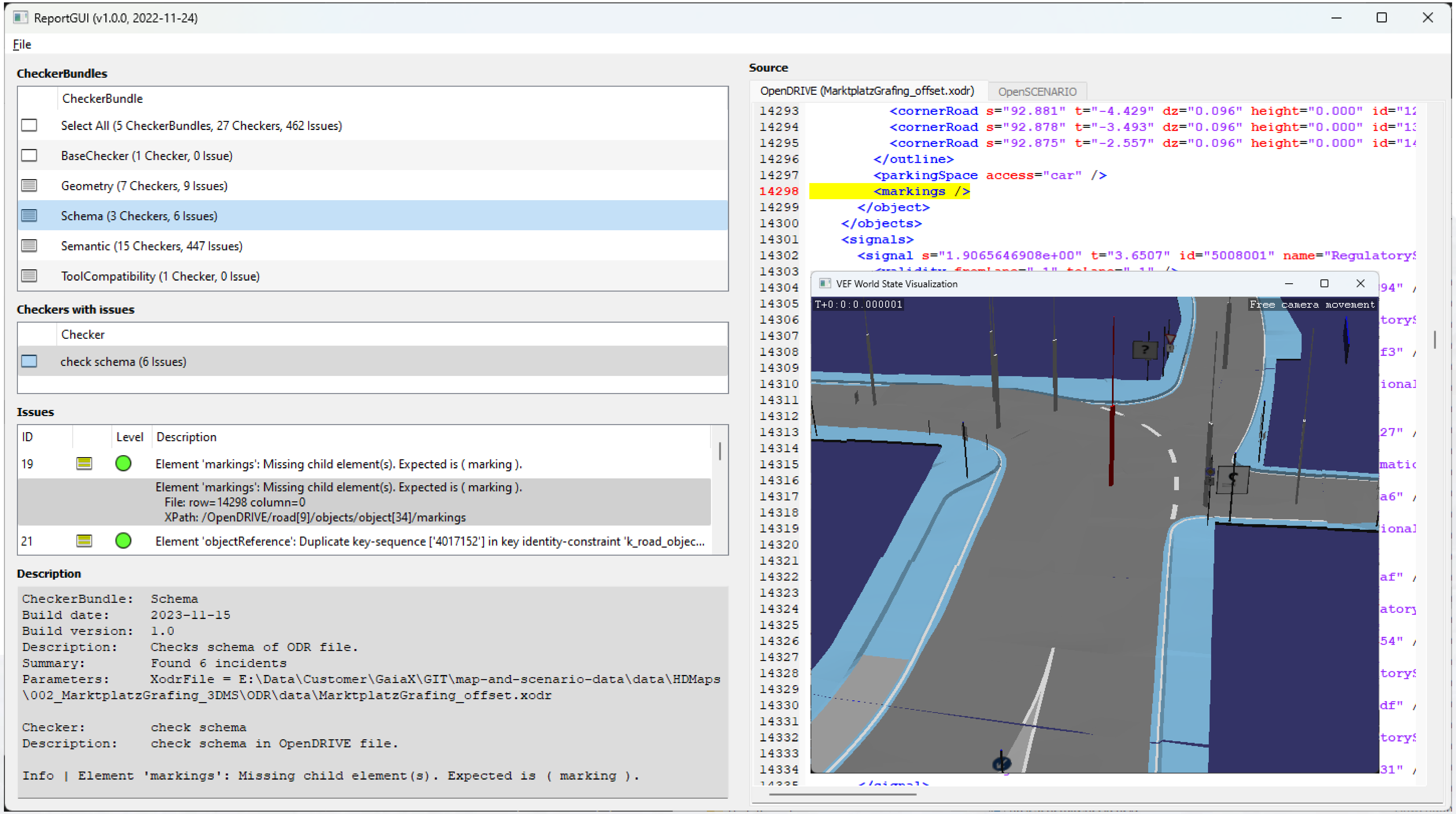


```python
import json

logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s.%(msecs)03d [%(levelname)5s-%(name)s] {%(module)s -> %(f
                    datefmt='%d/%m/%Y %H:%M:%S')
logging.getLogger(__name__).setLevel(logging.WARNING)


def main():
    parser = argparse.ArgumentParser(prog='main.py',
                                     description='Validates a given XML based OpenX file.')

    parser.add_argument('-o', '--output-directory', type=str, default='reports/', help='Path to
    parser.add_argument('-t', '--output-type', choices=['xqar', 'json', 'txt'], default='xqar',
    parser.add_argument('-a', '--addition-check-dirs', action='append', help='Additional directo
    parser.add_argument('-c', '--config', type=str, default='OpenValidator/ODR/config.json', hel
    parser.add_argument('INPUT_FILES', nargs='+', help='')

    args = parser.parse_args()

    # get output dir
    output_directory = Path(args.output_directory)
    if not output_directory.exists():
        logging.info(f'Provided output foler ({output_directory.absolute()}) does not exist. Cre
```

```
16/11/2023 13:36:58.391 [DEBUG-root] {validator -> run_checks} Loading checker {xodr.checks.tool_compatibility_che
cks.check_BMW_spider_roadType_vs_speed_limit}
16/11/2023 13:36:58.391 [ INFO-root] {check_BMW_spider_roadType_vs_speed_limit -> check_BMW_spider_roadType_vs_spe
ed_limit} check BMW Spider RoadType vs SpeedLimit
16/11/2023 13:36:58.392 [ INFO-root] {main -> main} write to reports\ValidSampleRoad-Test.xodr.xqar
PS E:\Data\Customer\GaiaX\GIT\map-and-scenario-data>
```

Live Demo

QChecker



ReportGUI (v1.0.0, 2022-11-24)

File

CheckerBundles

	CheckerBundle
□	Select All (5 CheckerBundles, 27 Checkers, 462 Issues)
□	BaseChecker (1 Checker, 0 Issue)
□	Geometry (7 Checkers, 9 Issues)
□	Schema (3 Checkers, 6 Issues)
□	Semantic (15 Checkers, 447 Issues)
□	ToolCompatibility (1 Checker, 0 Issue)

Checkers with issues

	Checker
□	check schema (6 Issues)

Issues

ID		Level	Description
19	▤	●	Element 'markings': Missing child element(s). Expected is (marking).
			Element 'markings': Missing child element(s). Expected is (marking). File: row=14298 column=0 XPath: /OpenDRIVE/road[9]/objects/object[34]/markings
21	▤	●	Element 'objectReference': Duplicate key-sequence ['4017152'] in key identity-constraint 'k_road_objec...

Description

```
CheckerBundle:  Schema
Build date:     2023-11-15
Build version:  1.0
Description:    Checks schema of ODR file.
Summary:        Found 6 incidents
Parameters:     XodrFile = E:\Data\Customer\GaiaX\GIT\map-and-scenario-data\data\HDMaps
\002_MarktplatzGrafing_3DMS\ODR\data\MarktplatzGrafing_offset.xodr

Checker:        check schema
Description:    check schema in OpenDRIVE file.

Info | Element 'markings': Missing child element(s). Expected is ( marking ).
```

Source

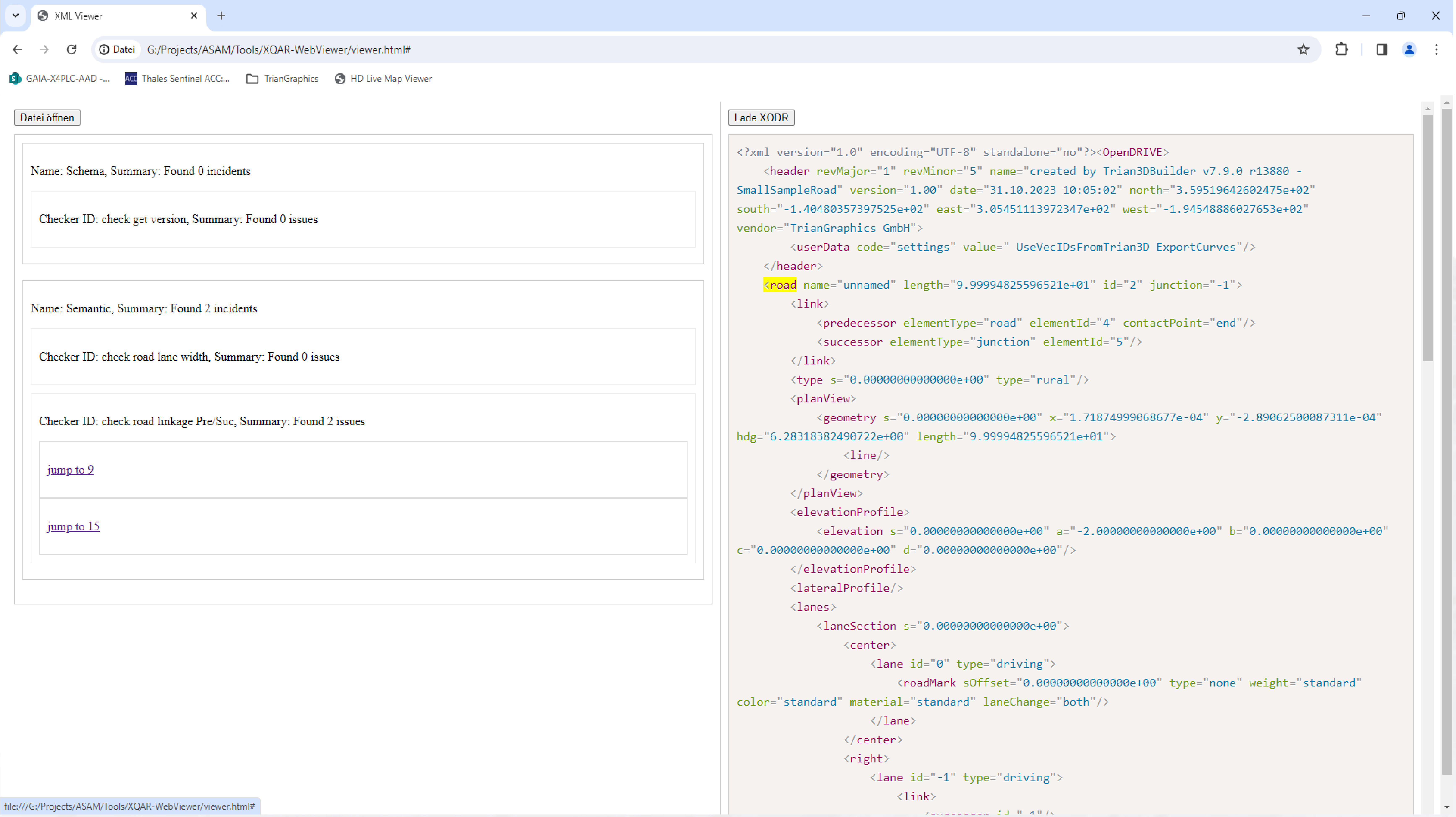
OpenDRIVE (MarktplatzGrafing_offset.xodr) | OpenSCENARIO

```
            <cornerRoad s="92.881" t="-4.429" dz="0.096" height="0.000" id="12
            <cornerRoad s="92.878" t="-3.493" dz="0.096" height="0.000" id="13
            <cornerRoad s="92.875" t="-2.557" dz="0.096" height="0.000" id="14
          </outline>
          <parkingSpace access="car" />
          <markings />
        </object>
      </objects>
      <signals>
        <signal s="1.9065646908e+00" t="3.6507" id="5008001" name="RegulatoryS
```

VEF World State Visualization

Live Demo

Webviewer



Name: Schema, Summary: Found 0 incidents

Checker ID: check get version, Summary: Found 0 issues

Name: Semantic, Summary: Found 2 incidents

Checker ID: check road lane width, Summary: Found 0 issues

Checker ID: check road linkage Pre/Suc, Summary: Found 2 issues

jump to 9

jump to 15

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><OpenDRIVE>
    <header revMajor="1" revMinor="5" name="created by Trian3DBuilder v7.9.0 r13880 -
SmallSampleRoad" version="1.00" date="31.10.2023 10:05:02" north="3.59519642602475e+02"
south="-1.40480357397525e+02" east="3.05451113972347e+02" west="-1.94548886027653e+02"
vendor="TrianGraphics GmbH">
        <userData code="settings" value=" UseVecIDsFromTrian3D ExportCurves"/>
    </header>
    <road name="unnamed" length="9.99994825596521e+01" id="2" junction="-1">
        <link>
            <predecessor elementType="road" elementId="4" contactPoint="end"/>
            <successor elementType="junction" elementId="5"/>
        </link>
        <type s="0.00000000000000e+00" type="rural"/>
        <planView>
            <geometry s="0.00000000000000e+00" x="1.71874999068677e-04" y="-2.89062500087311e-04"
hdg="6.28318382490722e+00" length="9.99994825596521e+01">
                <line/>
            </geometry>
        </planView>
        <elevationProfile>
            <elevation s="0.00000000000000e+00" a="-2.00000000000000e+00" b="0.00000000000000e+00"
c="0.00000000000000e+00" d="0.00000000000000e+00"/>
        </elevationProfile>
        <lateralProfile/>
        <lanes>
            <laneSection s="0.00000000000000e+00">
                <center>
                    <lane id="0" type="driving">
                        <roadMark sOffset="0.00000000000000e+00" type="none" weight="standard"
color="standard" material="standard" laneChange="both"/>
                    </lane>
                </center>
                <right>
                    <lane id="-1" type="driving">
                        <link>
                            <successor id="-1"/>
```

ASAM CheckerLib

Next Steps

Next Steps

Found Issues

- **Schema**
  - Incorrect key definition for Object References?
    - Object References can refer multiple times to Object with unique ID, and therefore Object Reference id is not unique!
    - OpenDrive 1.5 – 1.7
  - Old OpenDrive schema file not usable?
    - OpenDrive 1.1 – 1.2
    - Element 'OpenDRIVE': No matching global declaration available for the validation root.
  - Outline_cornerRoad and outline_cornerLocal should not have a key reference to outlineId
    - OpenDrive 1.5
    - Fixed already in 1.6-1.7
    - Element 'cornerLocal': No match found for key-sequence ['0'] of keyref 'r_road_objects_object_outline_cornerLocal’.
- **Examples** for OpenDrive 1.7
  - Ex_Railway-station
    - 2 xords – one (Ex_Railway_station.xodr) is in old OpenDrive 1.5
  - UC_Motorway-Exit-Entry
    - Lane validity for signals should not be “0”
    - E.g. UC_Motorway-Exit-Entry-DirectJunction.xord

Next Steps

Todo's

- Framework Todo's
  - Robustness
  - More flexible configuration files
    - Use configuration file from QChecker
    - Format specific, enable bundles/checks
    - Automatic registration of configurations from the checks themselves
- Documentation
  - Doxygen or AsciiDoc documentation in python scripts
  - What form and content?

- Checks
  - Current checks
    - must be reviewed
    - must be tested for different format versions
  - Example
    - Test all issue code paths
  - Further checks
    - ➢ Which Basic checks?
    - ➢ And for which older format versions?
      - not everyone supports newer versions
    - Use of the rule sections in the specifications
    - New features for new format versions
      - OpenDRIVE 1.8
      - OpenSCENARIO 1.3. XML
      - ➢ Where can I get sample data?
    - New checks could lead to errors at the data provider
      - Preparation time until publication so that data providers can correct this on their side

Next Steps

Todo’s

- QChecker
  - Implementation of 3D visualization
    - Use from esmini or libOpenDRIVE
  - Other Bundle Executables from other directories
  - GUI-Report Executable
    - for other text-based formats
    - Support for displaying large files (5 MB limit)
    - Drag&Drop
    - Refresh after loading new result file
  - Bundle philosophy
    - ➢ Bundle executables vs. Check folder?
    - ➢ Result file per bundle or total?
  - Report Tooling
    - ➢ How to deal with result files from different tools and similar categories/bundles?
      - Specification of tool and version

Next Steps

Questions

- How to deal with other validation tools?
  - **rtron** - https://github.com/tum-gis/rtron
  - Data / Tool Provider Tools
  - Integration of their test
    - for completeness, ease of use
  - or support of the result format
    - for flexibility, but complicated
- Attempt to be **Complete?**
  - ➢ possible at all or a goal of this project?
- Project activity vs. Opensource Initiative ?

THANKS FOR YOUR INTEREST

TRIANGRAPHICS
Intelligent Terrain Solutions

- **Contact**

✉ Mirco.Nierenz@triangraphics.de

www.TrianGraphics.com

www.Trian3DBuilder.com

www.youtube.com/user/TrianGraphics

www.linkedin.com/company/triangraphics-gmbh