Certification efforts in China

ASAM Technical Seminar 2023

Dr. Mian DAI
Technical Director of Intelligent and connected technology Co.,LTd.,
CATARC, China

March 22nd, 2023
Fürstenfeldbruck, Germany

C-ASAM

ASAM Association for Standardization of Automation and Measuring Systems

Motivation

Why do we need certification?

Motivation

Why do we need certification of ASAM OpenX standards

To ensure a high level of quality and consistency across the entire ecosystem of tools and systems

The adoption and use of ASAM OpenX standards

From **industry development** perspective

- Improving the utility of scenario-based validation
- In turn fosters
  greater interoperability and collaboration
  among tool vendors, sys integrator, and end users

From **technical** point of view

- Data libraries are complete and accurate
- Tools can effectively use these data
- Evaluation methods can be ported across different tools and platforms

Market needs: providing a standardized approach to certification

- Address **China-specific** **requirements** development & systems/tools application.
- Help to promote greater **efficiency**, **innovation**, and **competitiveness** in the automotive industry
- Expands **globally**

Objectives

What we want to achieve

Objectives

Promote the ASAM OpenX application, improve the practice of scenario-based validation

- Establishing **a common framework for certification**

  ASAM OpenX standards can be used with confidence across a wide range of applications and industries.

- Provides a **rigorous process for verifying**

  Implementations of ASAM OpenX standards meet the required quality and consistency standards

Business Benefits and Objectives

Increased Marketability for OEM/Tier1:
Highlight the chance for companies to stick out of the crowd by having certified repositories a competitive advantage in the market

Access to New Business:
Open up new markets, as assurance to potential customers

Future-proofing:
Continually design products and services to meet achieved certification

Reduced Costs:
By reducing the custom's integration and testing

Improved Reputation:
By demonstrating their commitment to quality and customer satisfaction

Technical Benefits and Objectives

Improved Quality:
meet the ASAM standards quality needs, with consistent performance and reliable results
improve the quality of available data repositories

Interoperability:
Provide with a means to test before delivery. P&S from diff vendors supporting without compatibility issues

Improved Collaboration:
BTW diff vendors, as common language and framework for communication & integration

Agreement on criteria:
To qualify certain repositories for certain use cases

Reduced Risk:
When introduce new standards,assurance of independent tests;Extensibility for function iteration on specific logic update

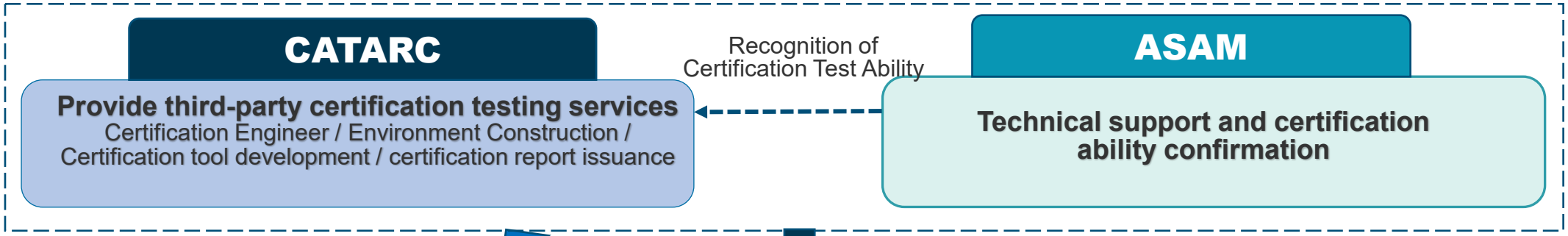
Objectives

Cooperation of C-ASAM, ASAM, CATARC, and Huacheng Certificate

ASAM and C-ASAM do not intend to provide certification by themselves but assisting the involved parties in setting up a process

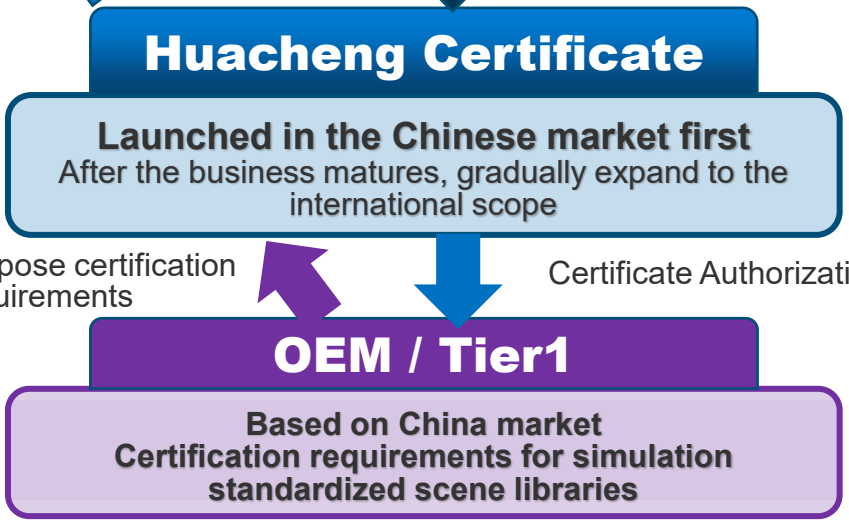
Step1□ ASAM OpenDRIVE 1.X

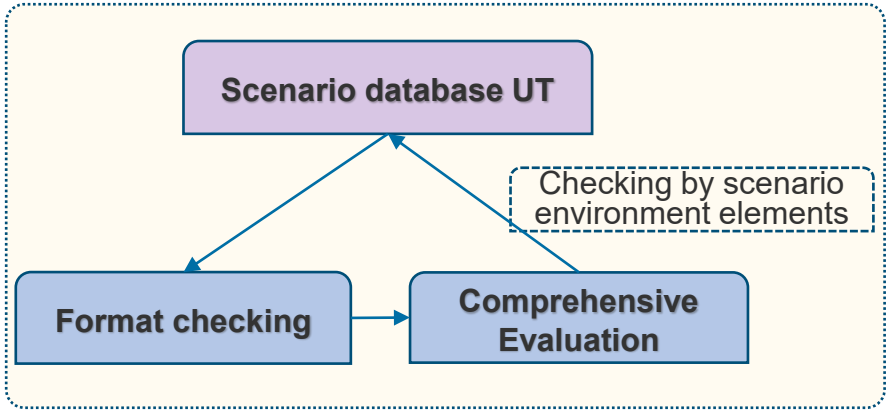
Step2□ ASAM OpenSCENARIO 1.X



Technical requirements
for certification testing

Certification test report





- Huge demand for verification of Scenario-based on China market
- High recognition of Chinese customers for the ASAM OpenX series standards
- Introduce landing business for scene library quality certification

Solution

How we will achieve our objectives

Overall concept

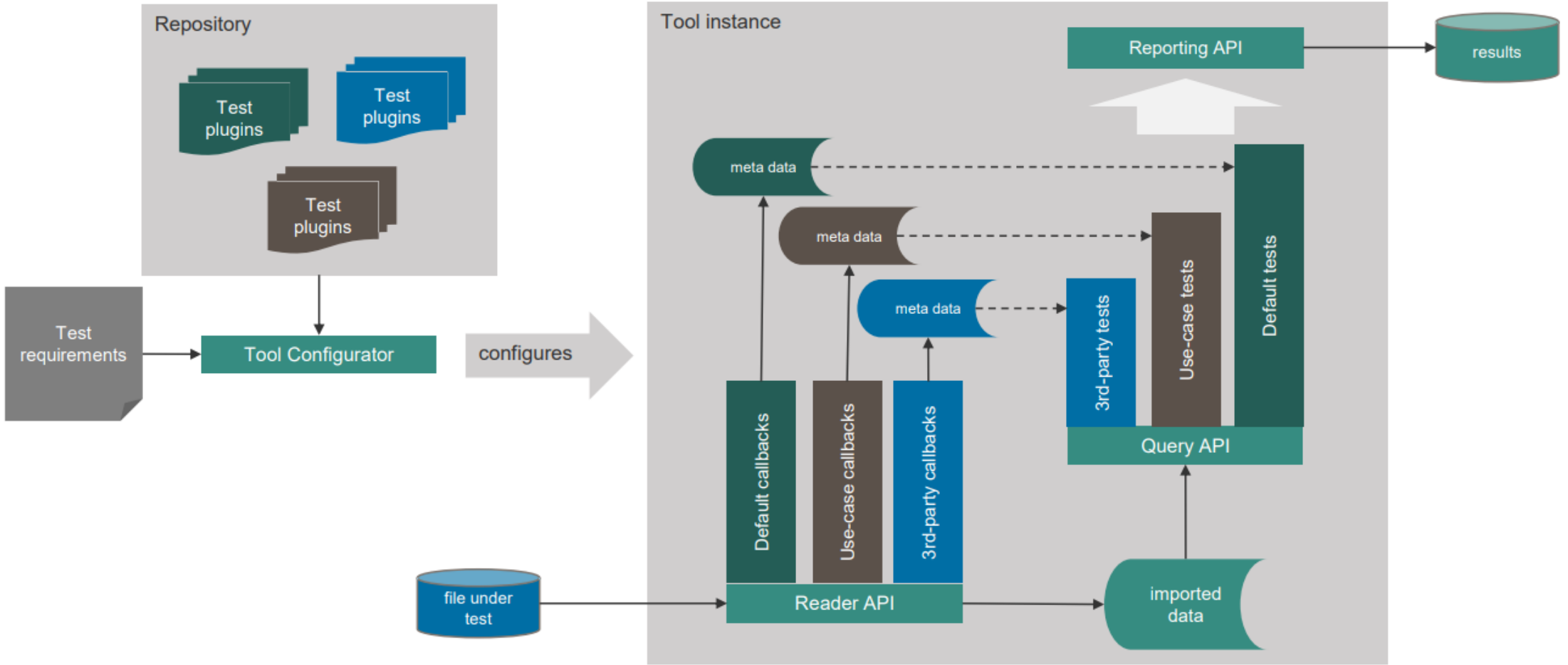
Data flow for checking the ASAM OpenDRIVE Library

Quality assurance procedure

Goals:
- no infringement of IP, copyrights, ownership etc.
- no manipulation of test results
- trustworthy results by a credible and trusted process

Checker tool structure

Goals:
- Maturity
- Various level of testing
- Extensibility



Results

What we have achieved

Results Stage 1: Formal Validation—XML syntax verification

- XML format and XML syntax check

Whether the XML syntax of the file is correct

Example ：

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  OpenDRIVE>          missing the start tag
3      <header revMajor="1" revMinor="4" name="" version="1" date="2022-07-15T13:27:43" north="5.3500582882808271e+2" south="-5.3
4          <geoReference><![CDATA[+proj=tmerc +lat_0=-0.0008005614997797679 +lon_0=118.5445870801937 +k=1 +x_0=0 +y_0=0 +datum=WG
5          <userData>
6              <vectorScene program="RoadRunner" version="R2022a Update 2 (1.4.2.b34d038e9e)"/>
7          </userData>
8      </header>
9  </OpenDRIVE>
```

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OpenDRIVE>
3      <header revMajor="1" revMinor="4" name="" version="1" date="2022-07-15T13:27:43" north="5.3500582882808271e+2" south="-5
4          <geoReference><![CDATA[+proj=tmerc +lat_0=-0.0008005614997797679 +lon_0=118.5445870801937 +k=1 +x_0=0 +y_0=0 +datum=
5          <userData>
6              <vectorScene program="RoadRunner" version="R2022a Update 2 (1.4.2.b34d038e9e)"/>
7
8      </header>          missing the ending tag
9  </OpenDRIVE>
```

verification results ：

```
PS C:\Users\ASUS\Desktop\xsd_schema>  c:; cd 'c:\Users\ASUS\Desktop\xsd_schema'; & 'C:\
.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '65333' '--' 'c:\User
syntax error:Start tag expected, '<' not found, line 2, column 1
```

```
PS C:\Users\ASUS\Desktop\xsd_schema>  c:; cd 'c:\Users\ASUS\Desktop\xsd_schema'; & 'C:\Users\ASUS\AppData\Local\Pro
.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '65424' '--' 'c:\Users\ASUS\Desktop\xml_validate-
syntax error:Opening and ending tag mismatch: userData line 5 and header, line 8, column 14
```

Results Stage 1: Formal Validation—XML syntax verification

Verification results：

```python
# -*- coding:utf-8 -*-
"""
usage:
    python xml_schema_check.py schema_file data_file
"""
# Import the etree module from the lxml library
from lxml import etree

# Import the XSD file and the XML file
schema_file = 'C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/opendrive_17_core.xsd'
data_file = 'C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr'

# Parse the XSD file and generate the Schema validator
schema_doc = etree.parse(schema_file)
schema = etree.XMLSchema(schema_doc)

# Verify XML syntax and Schema constraints, output errors locations and reasons
try:
    data = etree.parse(data_file)
    print(schema.validate(data))
    print('Schema verification errors:\n' + str(schema.error_log))
except etree.XMLSyntaxError as err:
    print('syntax error:' + err.msg)
```

问题　输出　调试控制台　终端　JUPYTER

```
.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '49467' '--' 'c:\Users\ASUS\Desktop\xml_validate-1.0\xml_validate-1.0\xml_test.py'
False
Schema verification errors:
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:5:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:221:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:236:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:254:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:275:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:300:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:328:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:368:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:421:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:426:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:898:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:946:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:992:0:ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:1062:0 ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:1067:0 ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:1551:0 ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
C:/Users/ASUS/Desktop/xml_validate-1.0/xml_validate-1.0/xml.xodr:1595:0 ERROR:SCHEMASV:SCHEMAV_CVC_COMPLEX_TYPE_4: Element 'userData': The attribute 'code' is required but missing.
```

the location of errors

the reason of errors

Results Stage 2: Data Consistency and use-case specific tests for ASAM OpenDRIVE

Data consistency tests

- Road reference lines
  - Primitives
    - The absolute position difference between the computed end of a primitive and the start position and orientation of the next element is below a tbd. threshold (e.g. 0.01mm)
    - The relative curvature difference between the end of a primitive and the start of the next primitive is below a tbd. threshold (e.g. 0.01%)
  - Elevation
    - The elevation entries within a reference line are consistent and there is no discontinuity in elevation
  - Superelevation / Crossfall etc.
    - see elevation
- Lanes
  - Borders
    - The absolute 3d positions of the left and right corners of a lane and its successor are within a tbd. distance (e.g. 1mm)
  - Direction
    - The direction of a lane and its successor does differ by more than a tbd. threshold (e.g. 0.001rad)
- Road and lane links
  - The IDs of elements given as successor and predecessor of roads and lanes do exist and are of the expected type
- Junction entries
  - All roads and lanes referenced in junction entries exist

Tests for use case „HIGHWAY“

- Lane curvature
  - No lane's curvature (measured along lane center) is above a tbd. threshold (for Germany, see Autobahnen | Bauformeln: Formeln online rechnen)
- Lane slope, vertical radius etc.
  - See „Lane curvature“
- Lane width
  - The width of each through lane is always above a tbd. value (e.g., 3.5m)
  - The variation of a lane's width is smooth and continuous

Tests for use case „PARKING“

- Occurrence
  - The database contains parking spaces and/or parking lanes
  - The parking spaces and lanes are adjacent to lanes which are not of type „motorway“
- Size
  - Parking spaces and/or lanes are wide and long enough to fit at least one passenger car (e.g., 2.50m x 5.00m)

Results Stage 2: Data Consistency Check — Road reference lines: Primitives

Code：

```python
# Parse xml file, generate dom tree and find all <road> nodes
def validate_opendrive(file_path):
    tree = ET.parse(file_path)
    root = tree.getroot()
    for road in root.findall('road'):
        validate_Road_reference_lines(road)
```

Create a DOM tree by parsing the OpenDRIVE file,
then extract all of the "road" nodes from it.

```python
def validate_Road_reference_lines(road):
    for i in range(len(road.findall('planView/geometry')) - 1):
        # extract the starting coordinate, length and starting angle of the one and the next road reference lines
        segment1 = road.findall('planView/geometry')[i]
        segment2 = road.findall('planView/geometry')[i + 1]
        s1 = float(segment1.get('s'))
        x1 = float(segment1.get('x'))
        y1 = float(segment1.get('y'))
        hdg1 = float(segment1.get('hdg'))
        length1 = float(segment1.get('length'))
        s2 = float(segment2.get('s'))
        x2 = float(segment2.get('x'))
        y2 = float(segment2.get('y'))
        hdg2 = float(segment2.get('hdg'))
        length2 = float(segment2.get('length'))
```

A "road" is made up of many "road reference line" segments.

According to its structure, each "road" is divided into a number of segments of road reference lines,

from which the starting location coordinate of the "road coordinate system" and the "absolute coordinate system," starting coordinates, start direction, and length value are extracted.

Results Stage 2: Data Consistency Check — Road reference lines: Primitives

Code ：

```
# the former road reference line is line
if segment1.findall('line'):
    # check if the end of one road reference line matches the start of the next
    if abs(np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2) - length1) > 1e-5 or abs(s2 - s1 - length1) > 1e-5:
        raise Exception("road reference lines are not continuously for road {0}".format(road.get('id')))
    # check if one road reference line connects smoothly with the next
    if abs(hdg2 - hdg1) > (1e-2) / 180 * math.pi:
        raise Exception("road reference lines are not smoothly for road {0}".format(road.get('id')))

# the former road reference line is arc
if segment1.findall('arc'):
    arc1 = segment1.findall('arc')
    curvature1 = float(arc1.get('curvature'))
    # check if the end of one road reference line matches the start of the next
    if abs((np.arcsin((np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2))/2 * curvature1) * 2) / (2 * math.pi) * (2 * math.pi / curvature1) - length1) > 1e-5  or abs(s2
        raise Exception("road reference lines are not continuously for road {0}".format(road.get('id')))
    # check if one road reference line connects smoothly with the next
    if abs(hdg2 - hdg1-(np.arcsin((np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2))/2 * curvature1) * 2)) > (1e-2) / 180 * math.pi:
        raise Exception("road reference lines are not smoothly for road {0}".format(road.get('id')))

# the former road reference line is spiral
if segment1.findall('spiral'):
    spiral1 = segment1.findall('spiral')
    curvStart1 = float(spiral1.get('curvStart'))
    curvEnd1 = float(spiral1.get('curvEnd'))
    # check if the end of one road reference line matches the start of the next
    if abs(s2 - s1 - length1) > 1e-5:
        raise Exception("road reference lines are not continuously for road {0}".format(road.get('id')))
```

Verify that the two road reference lines next to each other are connected and continuous.

Check the following in accordance with the previous road reference line's shape (line, arc, or spiral):
- whether it is continuous by comparing the starting coordinates of the subsequent segment to the end coordinates of the previous segment
- whether it is smooth by comparing the curvature of the subsequent segment's beginning curvature to the curvature of the previous segment's end point.

Results Stage 2: Data Consistency Check — Road reference lines: Primitives

Example

```
<road name="def1" length="6.1415926535897932e+02" id="1" junction="-1" rule="RHT">
    <link>
        <predecessor elementType="road" elementId="2" contactPoint="end"/>
        <successor elementType="road" elementId="2" contactPoint="start"/>
    </link>
    <planView>
        <geometry s="0.0000000000000000e+00" x="0.0000000000000000e+00" y="0.0000000000000000e+00" hdg="0.0000000000000000e+00" length="3.0000000000000000e+02">
            <line/>
        </geometry>
        <geometry s="3.0000000000000000e+02" x="0.0000000000000000e+02" y="0.0000000000000000e+00" hdg="0.0000000000000000e+00" length="3.1415926535897932e+02">
            <arc curvature="1.0000000000000000e-02"/>
        </geometry>
    </planView>
```

Continuous splicing is impossible because the succeeding segment's starting

Verification results：

```
28        raise Exception("road reference lines are not continuously for road {0}".format(road.get('id')))

发生异常: Exception ×
road reference lines are not continuously for road 1
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 28, in validate_Road_reference_lines
    raise Exception("road reference lines are not continuously for road {0}".format(road.get('id')))
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 59, in validate_opendrive
    validate_Road_reference_lines(road)
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 62, in <module>
    validate_opendrive('default_road.xodr')
Exception: road reference lines are not continuously for road 1
```

Results Stage 2: Data Consistency Check — Road reference lines: Primitives

Example

```
<road name="def1" length="6.1415926535897932e+02" id="1" junction="-1" rule="RHT">
    <link>
        <predecessor elementType="road" elementId="2" contactPoint="end"/>
        <successor elementType="road" elementId="2" contactPoint="start"/>
    </link>
    <planView>
        <geometry s="0.0000000000000000e+00" x="0.0000000000000000e+00" y="0.0000000000000000e+00" hdg="0.0000000000000000e+00" length="3.0000000000000000e+02">
            <line/>
        </geometry>
        <geometry s="3.0000000000000000e+02" x="3.0000000000000000e+02" y="0.0000000000000000e+00" hdg="1.0000000000000000e+00" length="3.1415926535897932e+02">
            <arc curvature="1.0000000000000000e-02"/>
        </geometry>
```

The succeeding segment's starting curvature is incorrect, making a smooth splicing impossible.

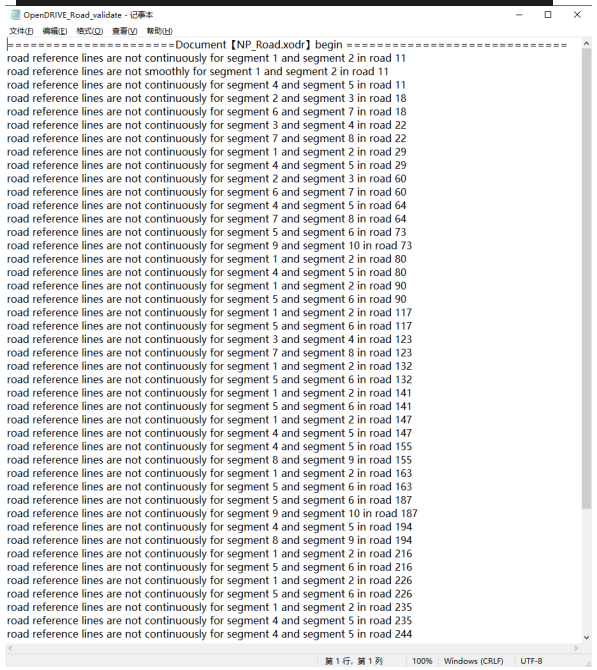
Verification results：

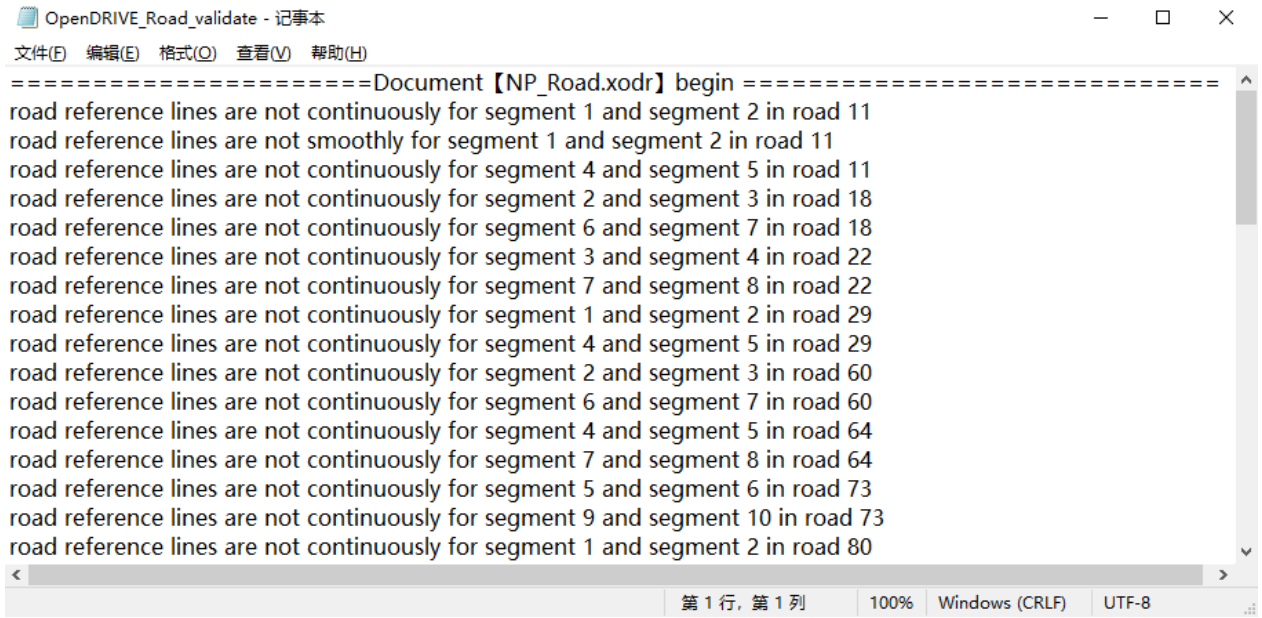
```
31        raise Exception("road reference lines are not smoothly for road {0}".format(road.get('id')))

发生异常: Exception ×
road reference lines are not smoothly for road 1
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 31, in validate_Road_reference_lines
    raise Exception("road reference lines are not smoothly for road {0}".format(road.get('id')))
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 59, in validate_opendrive
    validate_Road_reference_lines(road)
  File "C:\Users\ASUS\Desktop\xml\pyopendrive-test.py", line 62, in <module>
    validate_opendrive('default_road.xodr')
Exception: road reference lines are not smoothly for road 1
```

Results Stage 2: Data Consistency Check — Road reference lines: Primitives

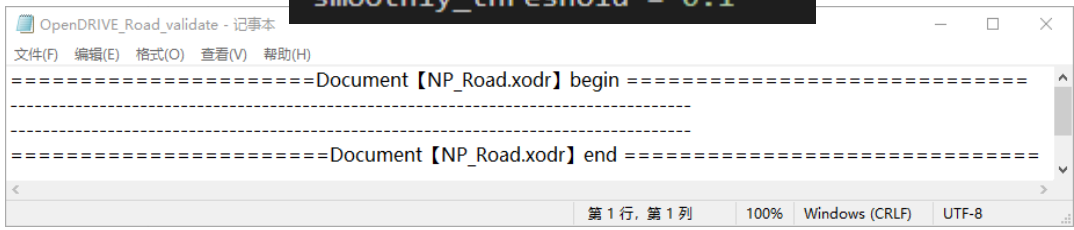
- The results can be output in a report file automatically
- Outputting the specific location where discontinuous or not smooth occurs
- Continuity and smoothness threshold settings proposal by user defined

```
continuity_threshold = 1e-5
smoothly_threshold = 0.01
```






```
continuity_threshold = 1e-3
smoothly_threshold = 0.1
```



Results Stage 2: Data Consistency Check — Road reference lines: Elevation

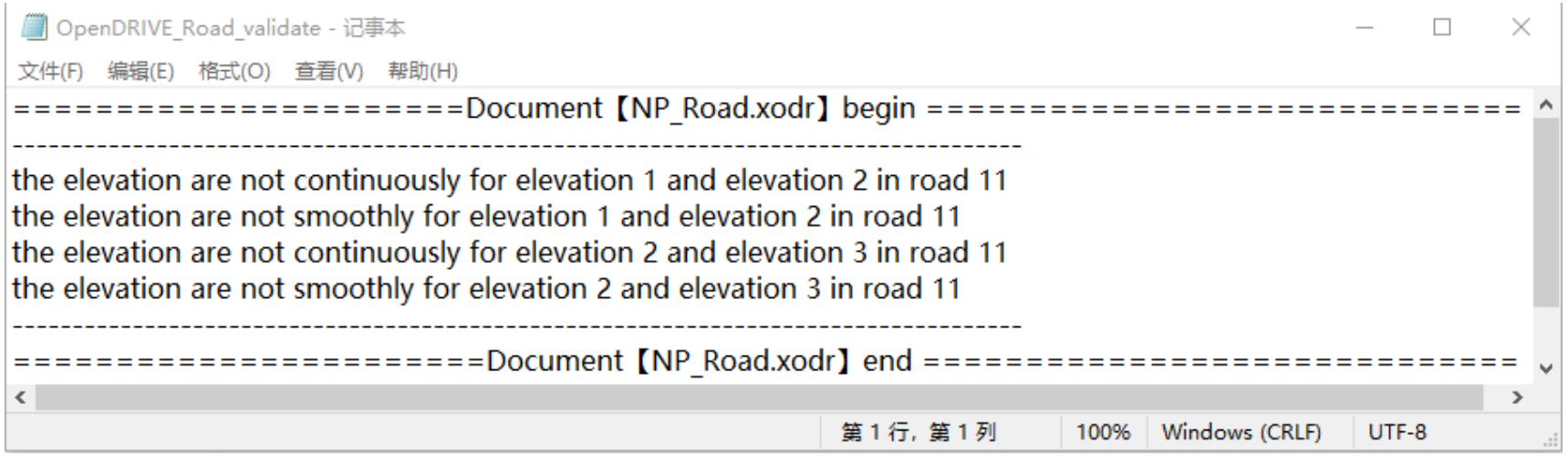
- Since ‘s’ does not change by the elevation value, it is unnecessary to consider the length of the road when check the elevation
- Only the continuity and smoothness of the elevation in the longitudinal segment along the road direction.

```python
# check if elevation entries within a reference line are consistent and there is no discontinuity
def validate_elevation(road,text_file):
    for i in range(len(road.findall('elevationProfile/elevation')) - 1):
        # extract attributes of the elevation element
        elevation1 = road.findall('elevationProfile/elevation')[i]
        elevation2 = road.findall('elevationProfile/elevation')[i + 1]
        s1 = float(elevation1.get('s'))
        a1 = float(elevation1.get('a'))
        b1 = float(elevation1.get('b'))
        c1 = float(elevation1.get('c'))
        d1 = float(elevation1.get('d'))
        s2 = float(elevation2.get('s'))
        a2 = float(elevation2.get('a'))
        b2 = float(elevation2.get('b'))
        c2 = float(elevation2.get('c'))
        d2 = float(elevation2.get('d'))

        continuity_threshold = 1e-3
        smoothly_threshold1 = 0.01

        # check if the elevation are no discontinuity
        if abs((a1 + b1 * (s2 - s1) + c1 * (s2 - s1) ** 2 + d1 * (s2 - s1) ** 3) - a2) > continuity_threshold:
            print("the elevation are not continuously for elevation {0} and elevation {1} in road {2}".format(i+1, i+2, road.get('id')))
            text_file.write("the elevation are not continuously for elevation {0} and elevation {1} in road {2}\n".format(i+1, i+2, road.get('id')))
        # check if the elevation are smoothly
        if abs((3 * d1 * s2 ** 2 + 2 * c1 * s2 + b1) - b2) > smoothly_threshold1:
            print("the elevation are not smoothly for elevation {0} and elevation {1} in road {2}".format(i+1, i+2, road.get('id')))
            text_file.write("the elevation are not smoothly for elevation {0} and elevation {1} in road {2}\n".format(i+1, i+2, road.get('id')))
```

Results Stage 2: Data Consistency Check — Road reference lines: Elevation

- Modifying the evaluation value, and the errors are reported with ID.
（slope needs to be considered， thd e.g.1%）also user defined

```
  </road>
  <road name="Road 11" length="3.90058076284312 85e+1" id="11" junction="9">
      <link>
<elevationProfile>
    <elevation s="0.0000000000000000e+0" a="0.0000000000000000e+0" b="0.0000000000000000e+0" c="0.0000000000000000e+0" d="0.0000000000000000e+0"/>
    <elevation s="8.2006786604429180e+0" a="2.0000000000000000e+0" b="2.0000000000000000e+0" c="0.0000000000000000e+0" d="0.0000000000000000e+0"/>
    <elevation s="1.1917247092280604e+1" a="0.0000000000000000e+0" b="0.0000000000000000e+0" c="0.0000000000000000e+0" d="0.0000000000000000e+0"/>
```



Results Stage 2: Data Consistency Check — Road reference lines: Super-elevation check

- Super-elevation is defined as the slope angle of the road along the t direction
- It is necessary to verify whether the slope angles of two adjacent segments are the same.

```
</road>
<road name="Road 11" length="3.9005807628431285e+1" id="11" junction="9">
    <link>
```

OpenDRIVE_Road_validate - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

==========Document【NP_Road.xodr】begin ==========

---

the superelevation are not continuously for superelevation 1 and superelevation 2 in road 11

the superelevation are not continuously for superelevation 2 and superelevation 3 in road 11

==========Document【NP_Road.xodr】end ==========

第 1 行，第 1 列 100% Windows (CRLF) UTF-8

Results Stage 2: Data Consistency Check — Road and Lane links

- To check the existence of the predecessor and successor
- There is no Direction in the Lane attribute.

code：

```
def verify(targets, roadId, str, elementId) :
    flag = True
    for target in targets:
        if target.hasAttribute("id") and target.getAttribute("id") == elementId :
            flag = False
            #print ("id: %s的道路中%s有问题%s" % (roadId, str, flag))
    if flag :
        print ("Error (T_T): %s of road %s does not exist in this OpenDRIVE!!!\n " % (str,roadId))
        text_file.write("Error (T_T): %s of road %s does not exist in this OpenDRIVE!!!\n " % (str,roadId))

# 使用minidom解析器打开 XML 文档
DOMTree = xml.dom.minidom.parse("NP_Road.xodr")
openDRIVE = DOMTree.documentElement
#if openDRIVE.hasAttribute("name"):
#   print "road element : %s" % openDRIVE.getAttribute("name")

Folder = 'xml1'
file_path = 'NP_Road.xodr'
text_path = 'OpenDRIVE_Road&Lane link_validate.txt'
text_file = open(text_path, mode='w', encoding='UTF-8')
print('====================Document【' + file_path + '】begin ============================')
text_file.write('====================Document【' + file_path + '】begin ============================\n')
```

```
# 在集合中获取所有road
roads = openDRIVE.getElementsByTagName("road")
# 在集合中获取所有junction
junctions = openDRIVE.getElementsByTagName("junction")
for road in roads:
    if road.hasAttribute("id"):
      roadId = road.getAttribute("id")
      link = road.getElementsByTagName('link')[0]
      if link.getElementsByTagName('predecessor') != []  :
          predecessor = link.getElementsByTagName('predecessor')[0]
          elementId = predecessor.getAttribute('elementId')
          if predecessor.getAttribute('elementType') == 'junction' :
              verify(junctions, roadId, 'predecessor', elementId)
          elif predecessor.getAttribute('elementType') == 'road' :
              verify(roads, roadId, 'predecessor', elementId)
      if link.getElementsByTagName('successor') != [] :
          successor = link.getElementsByTagName('successor')[0]
          elementId = successor.getAttribute('elementId')
          if successor.getAttribute('elementType') == 'junction' :
              verify(junctions, roadId, 'successor', elementId)
          elif successor.getAttribute('elementType') == 'road' :
              verify(roads, roadId, 'predecessor', elementId)

print('====================Document【' + file_path + '】end ============================\n')
text_file.write('====================Document【' + file_path + '】end ============================\n\n')
```

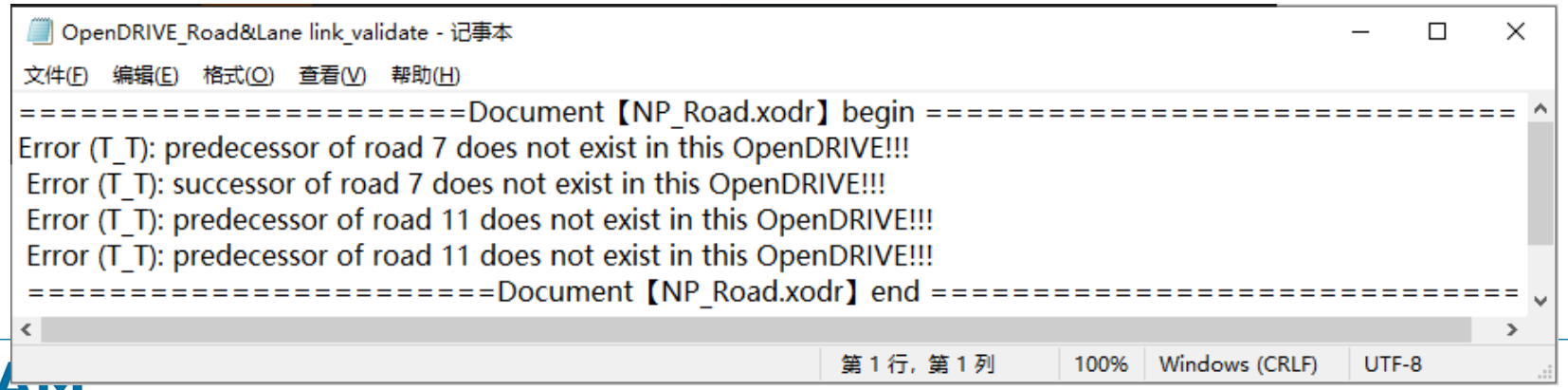
Results Stage 2: Data Consistency Check — Road and Lane links

Example：

```
<road name="Road 7" length="3.9913273161942402e+2" id="7" junction="-1">
    <link>
        <predecessor elementType="junction" elementId="8800"/>
        <successor elementType="junction" elementId="9000"/>
    </link>
```

```
<road name="Road 11" length="3.9005807628431285e+1" id="11" junction="9">
    <link>
        <predecessor elementType="road" elementId="8000" contactPoint="start"/>
        <successor elementType="road" elementId="2000" contactPoint="start"/>
    </link>
```

Verification results：



Next steps

We are not yet done…

Next steps

To complete the first certificate phase

- Use case for functions
- Checking codes integration with executable installation locally;
- Automated inspection and report generation;
- Analysis report templates based on mandatory and user-defined inspections

Example: Use case— Highway Pilot in China

To check:

- Road curvature allowed by China:
In plain and hilly areas, curve radius of HW curve is >= 650 meters
- lane slope:
The maximum slope of expressways is 5%, and the recommended maximum slope in snowy and cold areas is<= 6%.
- lane width: 3.75m in China, with continuity and smooth checking

Technical goals to achieve in the future

- OpenDrive --> SCENARIO
- Standardization, formalization with Customization
- Automated result return, compression and decoding

ASAM members worldwide benefit from the certificate activities

Demonstrating compliance with ASAM standards

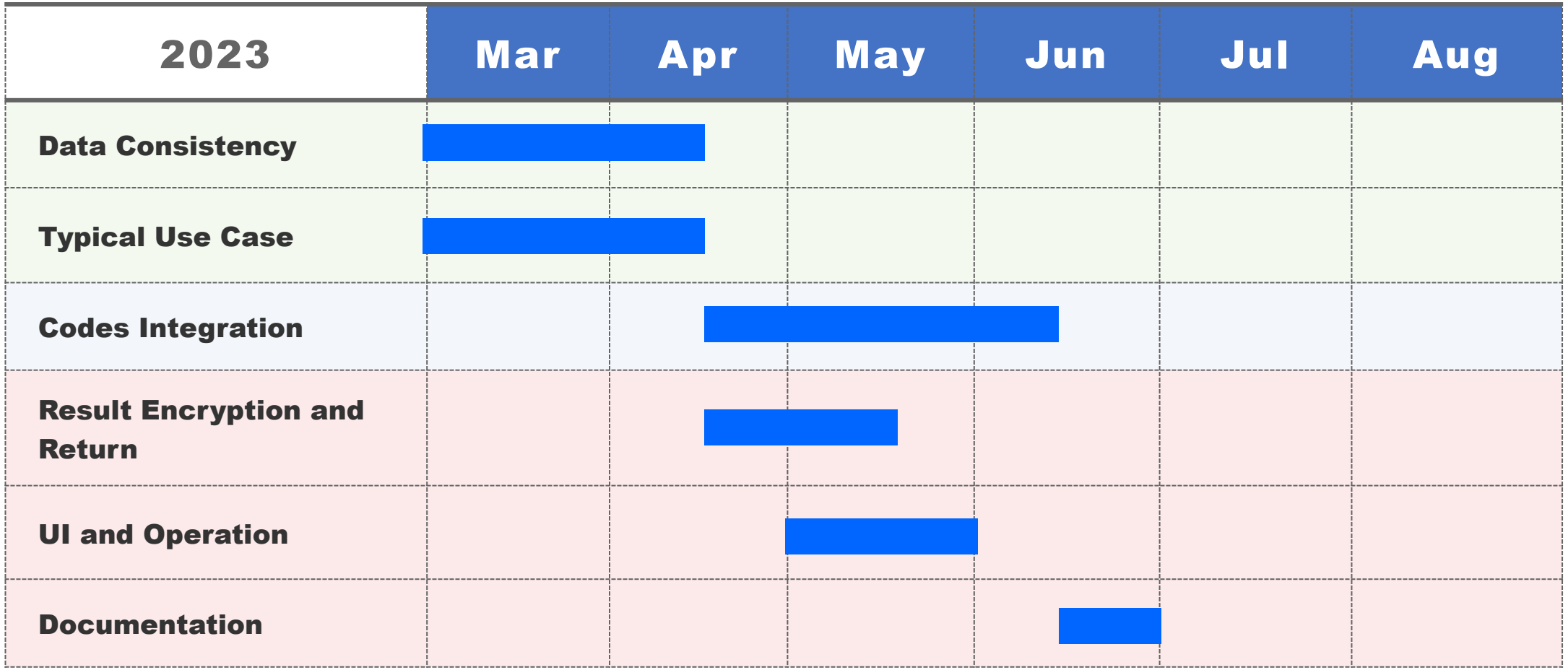
Improving brand recognition

Enhancing customer trust

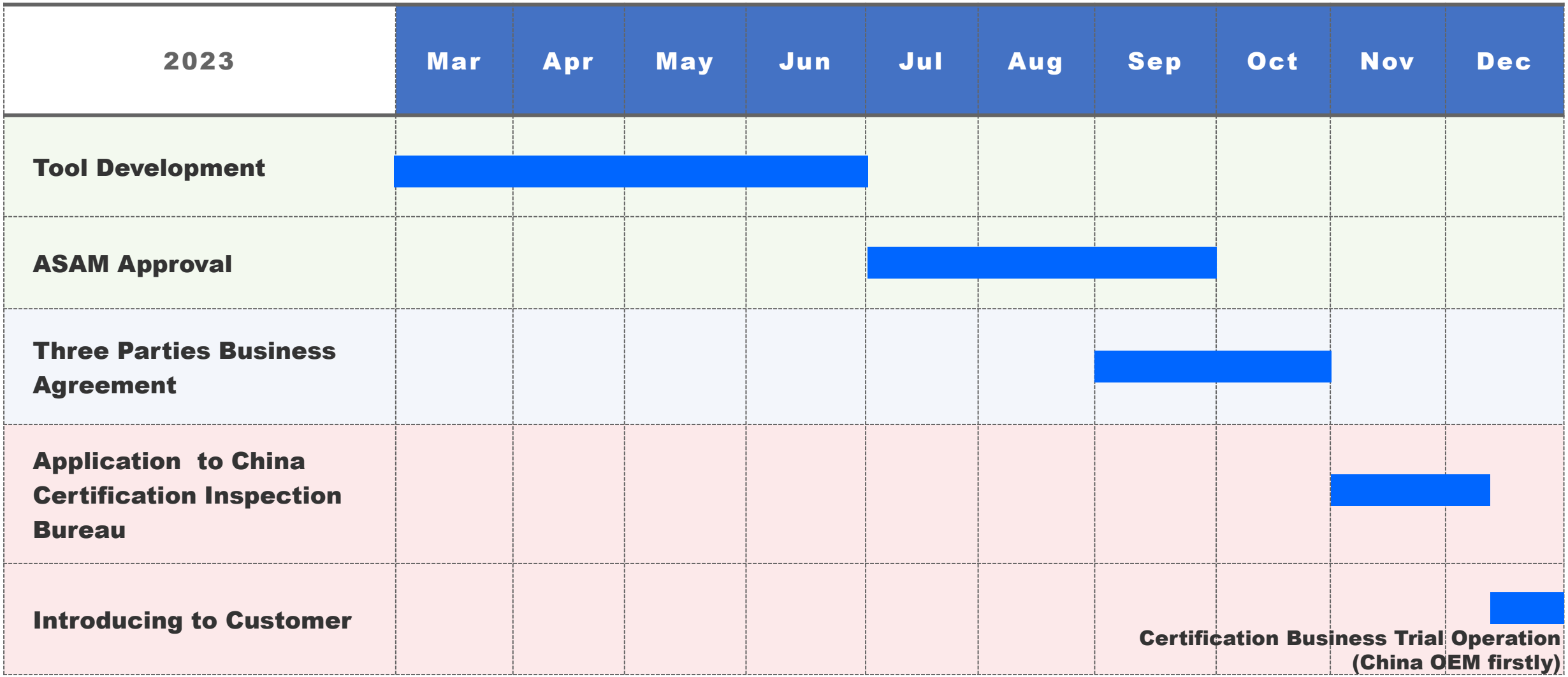
Encouraging innovation

Translate into a competitive advantage for OEM and vendors who offer certified P&S

Next steps: Tool Development



Timeline for introducing the certification in the Chinese market



Thank you for your attention!

Dr. Mian DAI
Technical Director of Intelligent and connected technology Co.,LTd.,
CATARC, China
email: daimian@catarc.ac.cn