High-performance serving of large-scale OpenDRIVE datasets using standardized GIS technology

6th Symposium Driving Simulation, 2020-11-05, Virtual Event

Michael Scholz

German Aerospace Center (DLR)

Institute of Transportation Systems

German Aerospace Center

Research institutes

- Aeronautics
- Space
- Energy
- Transportation
- Security
- Digitalisation

Space administration

Project management agency

Institute of Transportation Systems

Key facts
- In Berlin and Brunswick
- Around 220 employees

Research fields
- Automotive
- Railway systems
- Traffic management
- Multi-modal and public transport

Area of work
- Fundamental research
- Conception and strategy development
- Prototyping

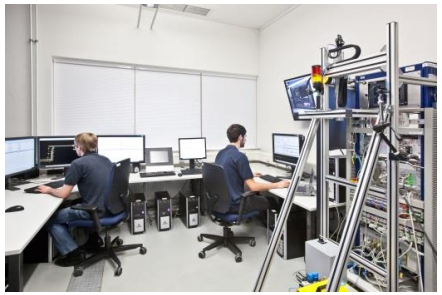
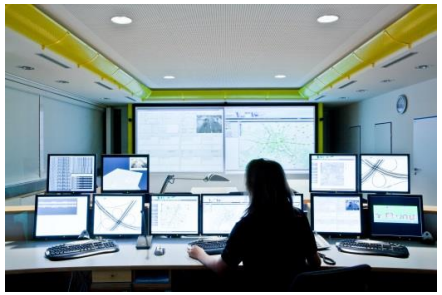
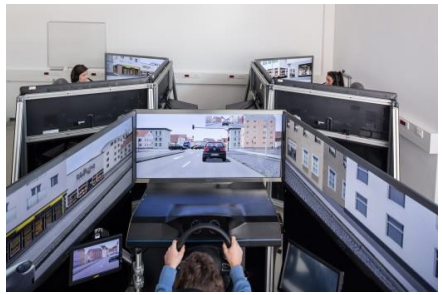
Our research infrastructure ...



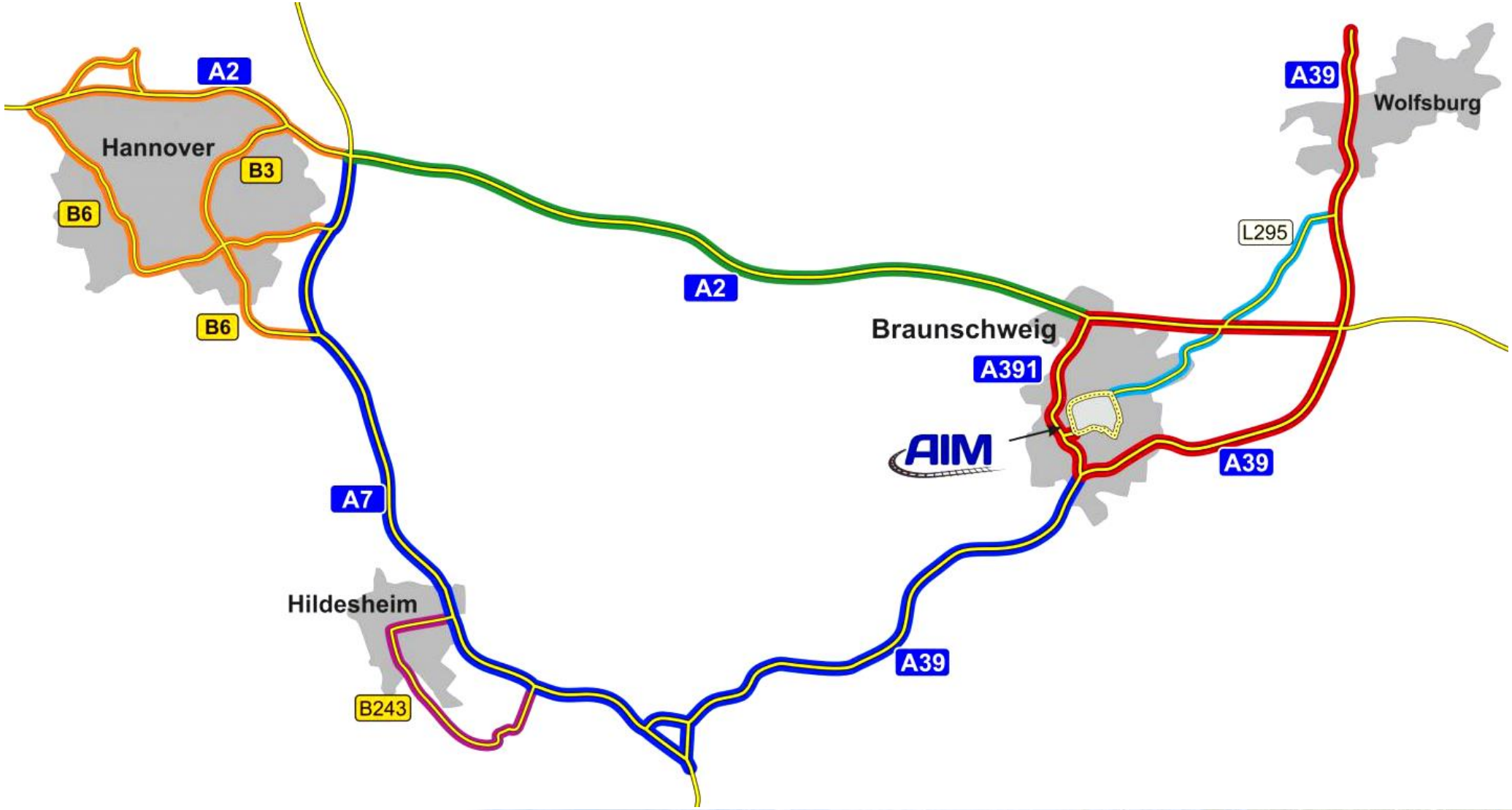






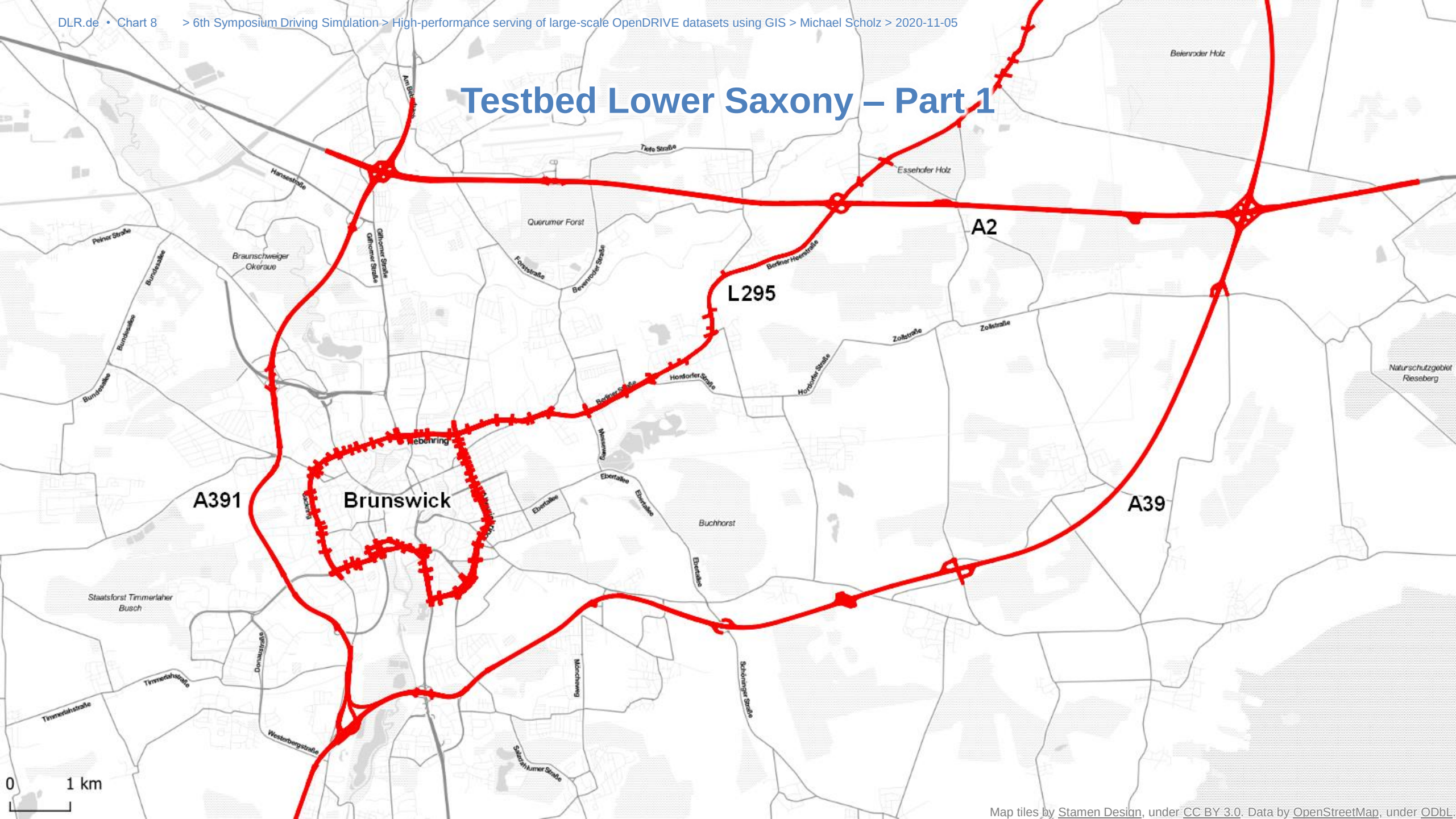






... and our Testbed of Lower Saxony

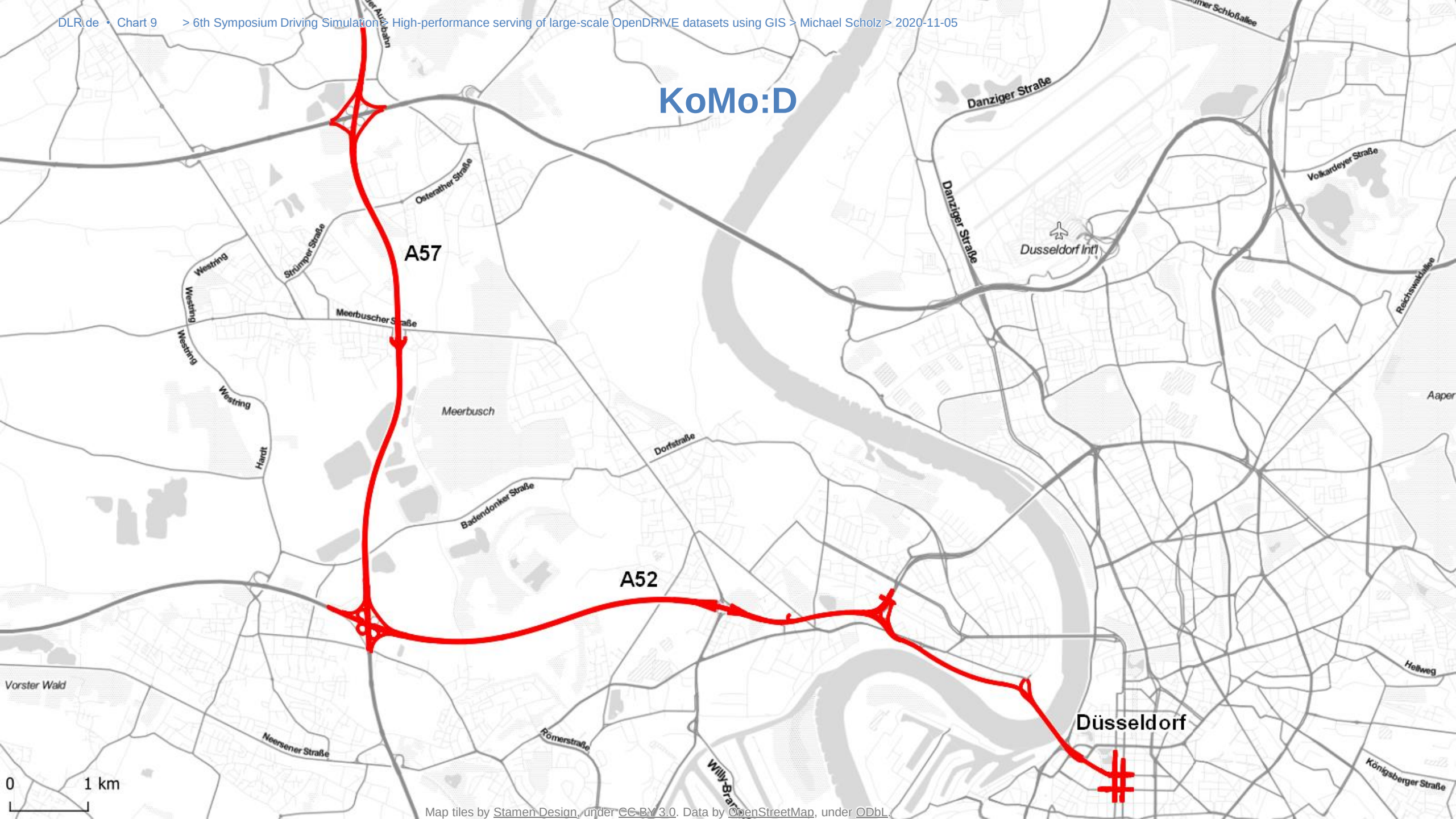


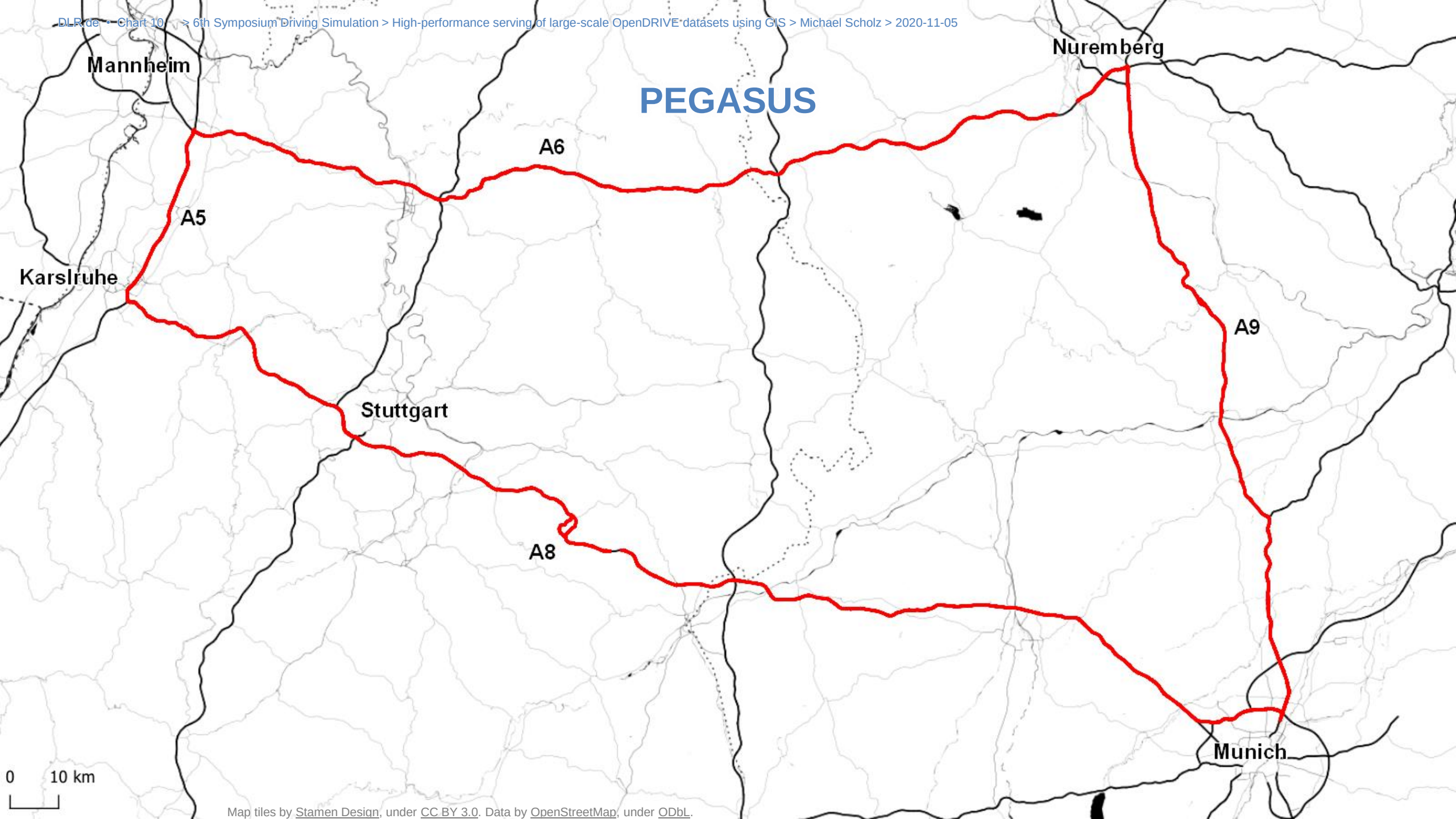
Challenges

Testbed Lower Saxony – Part 1



KoMo:D





What is desirable?

- Consistent **management** of such *heterogeneous* datasets
- Fast **access** and data browsing for different users
- Simple **interface**/API
- Easy **snippet extraction** from whole datasets

OpenDRIVE over time

Initial scope of application

- Fast prototyping of simulation tracks
  → Artificial/imaginary test data
- Restricted (small) spatial extent
- High modelling detail with visual properties
  → 3D rendering
- Simple, continuous geometry definition
  → Smooth road course
- Real-time processing capability

Current and future trends

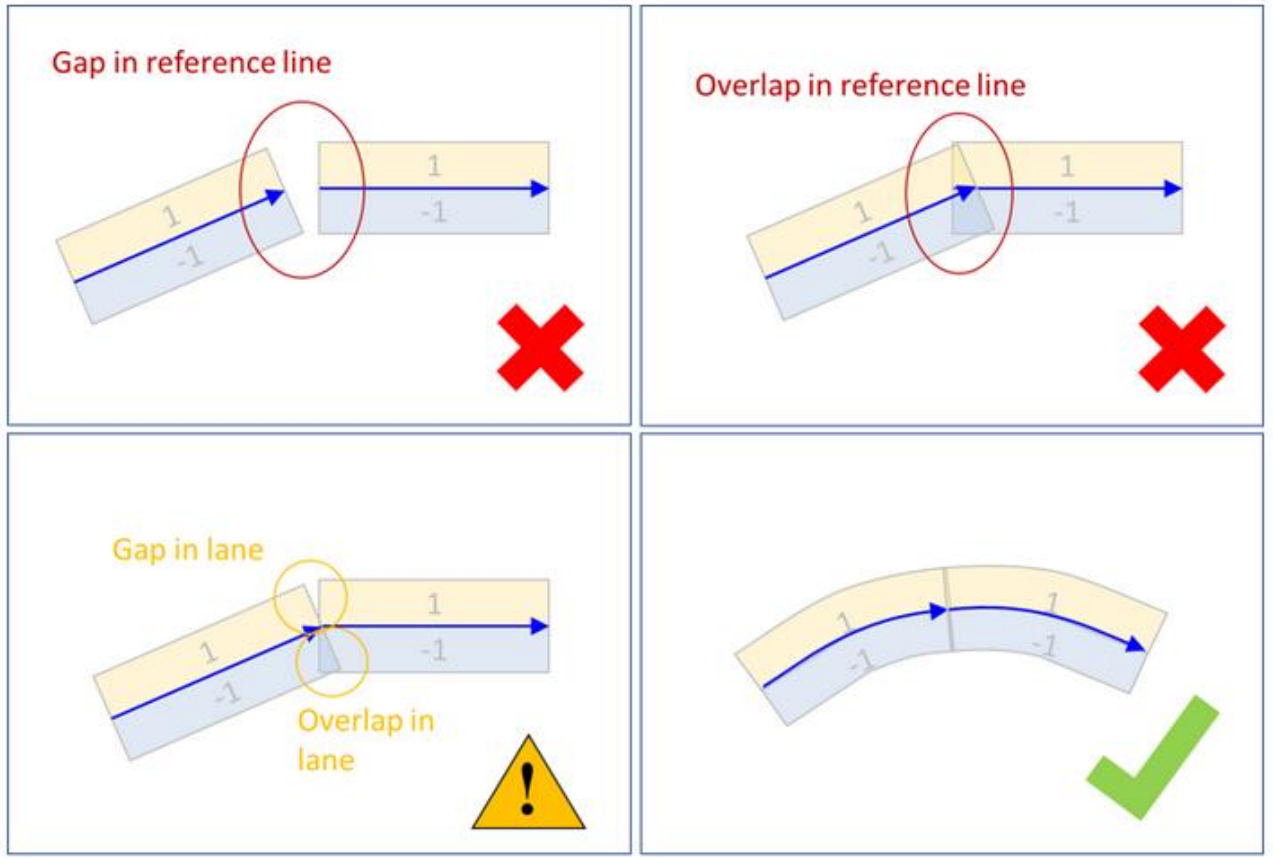
- Real-world data
  - From motorways over
  - inner cities to
  - multi-level parking decks
- Data updates and network merging
- Increasing spatial extents
- “From simulation into the car”
  → Electronic horizon, rejection of styling elements
- Linkage to supplementary environmental data

OpenDRIVE is bad for

Extracting/editing

- Because you need tools which are
  - commercial (money, money, money),
  - complicated,
  - inflexible,
  - ugly.

- Because there is no server-based solution “as a service”



see ASAM OpenDRIVE Format Specification, v. 1.6.0

OpenDRIVE is bad for

Data exchange

- Because of strong data model hierarchy and element cross-references

```
<road name="Boulevard of Rock" length="66.6"
    <link>
        <predecessor elementType="junction"
        <successor elementType="junction" el
    </link>
```

```
<signal s="0" t="0" id="1337"
 country="LV-426" subtype="-1"
    <laneValidity fromLane="1"
</signal>
```

```
<lanes>
    <laneSection s="0">
        <left>
            <lane id="3" type="border"
                <link>
                    <successor id="3"/>
                </link>
```

```
<junction name="ne Kreuzung halt" id="1234">
    <connection id="0" incomingRoad="1" connectingRoad="2"
        <laneLink from="-7" to="-7"/>
        <laneLink from="-6" to="-6"/>
        <laneLink from="-5" to="-5"/>
        <laneLink from="-4" to="-4"/>
```

OpenDRIVE is bad for

Data exchange

- Because of strong data model hierarchy and element cross-references
- Because small data snippets quickly result in millions of lines of *text*

```
18        </geometry>
19        <geometry s="3.11563112412e+02"
          "4.33321408575le+00" length="2.07
20          <paramPoly3 aU="-0.000000000000
            "-1.52262317471le-09" aV="0.000
             dV="1.09384152309e-08" pRange=
21        </geometry>
22        <geometry s="5.19272894392e+02"
          "4.31293371250le+00" length="1.03
23          <paramPoly3 aU="-0.000000000000
            "-1.08648946127e-09" aV="0.000
```

eXtensible Markup Language file | length : 322.530.389 lines : 2.497.443

Subset extraction is not trivial

Solving “the problem” in three steps

1. Make OpenDRIVE data GIS-able
2. Deploy GIS data in spatial database
3. Publish as RESTful web service

GIS? Why?

- Well-established **standards** around for 15+ years (OGC)
- A super-huge **community** of which all automotive guys can just dream about
- They know how to **handle huge data**
- Broad tool support, also **for free** (open source software)
- Some components offer **standardised services**
- Native workflow with cadastral data, CAD, Road2Simulation, Lanelet/2, …

GIS? Why?

- Well-established **standards** around for 15+ years (OGC)
- A super-huge **community** of which all automotive guys can just dream about
- They know how to **handle huge data**
- Broad tool support, also **for free** (open source software)
- Some components offer **standardised services**
- Native workflow with cadastral data, CAD, Road2Simulation, Lanelet/2, …

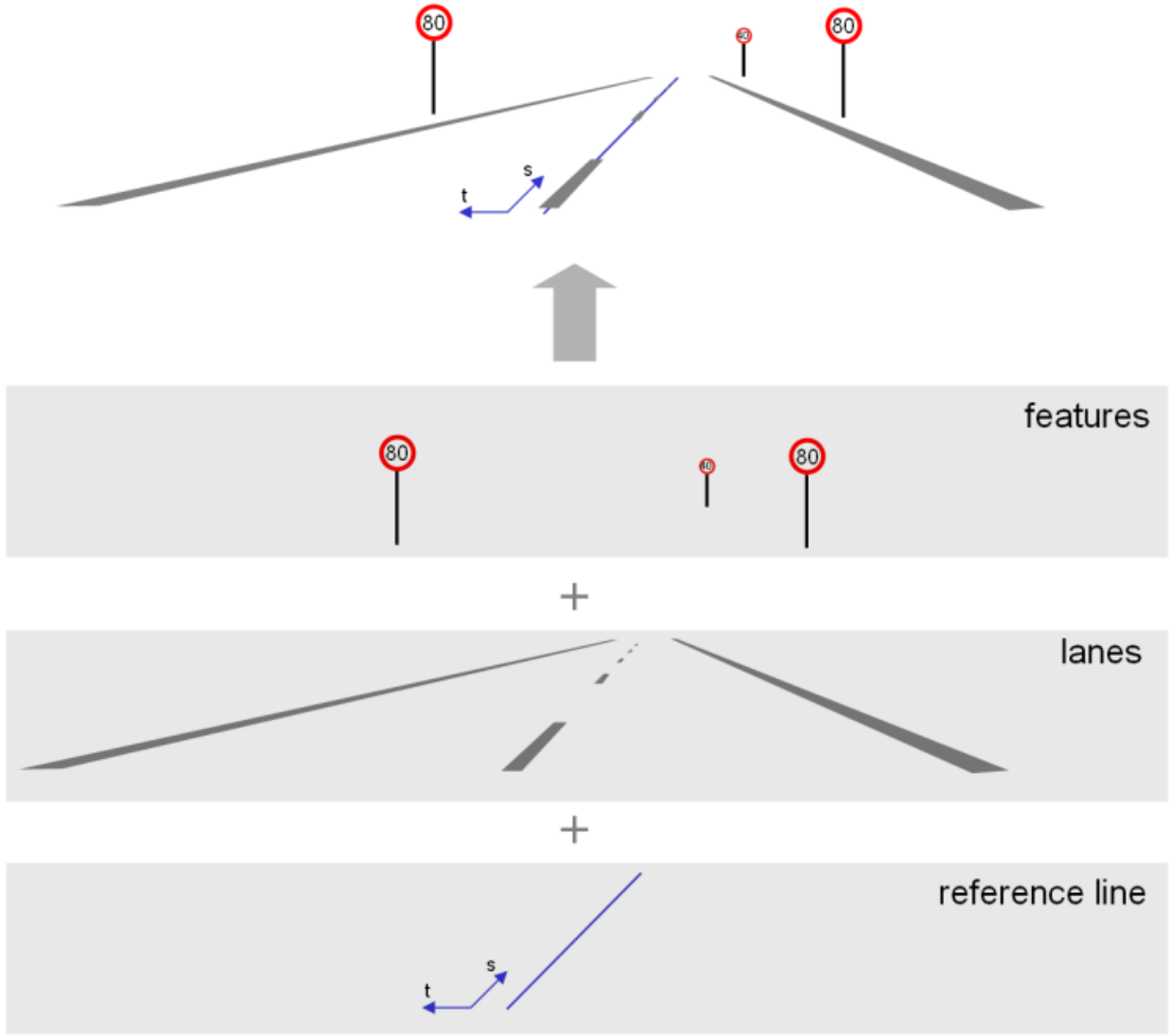
“Don’t re-invent the wheel”

Solving “the problem” in three steps

1. Make OpenDRIVE data GIS-able
2. Deploy GIS data in spatial database
3. Publish as RESTful web service

Make OpenDRIVE data GIS-able
Geometry basics

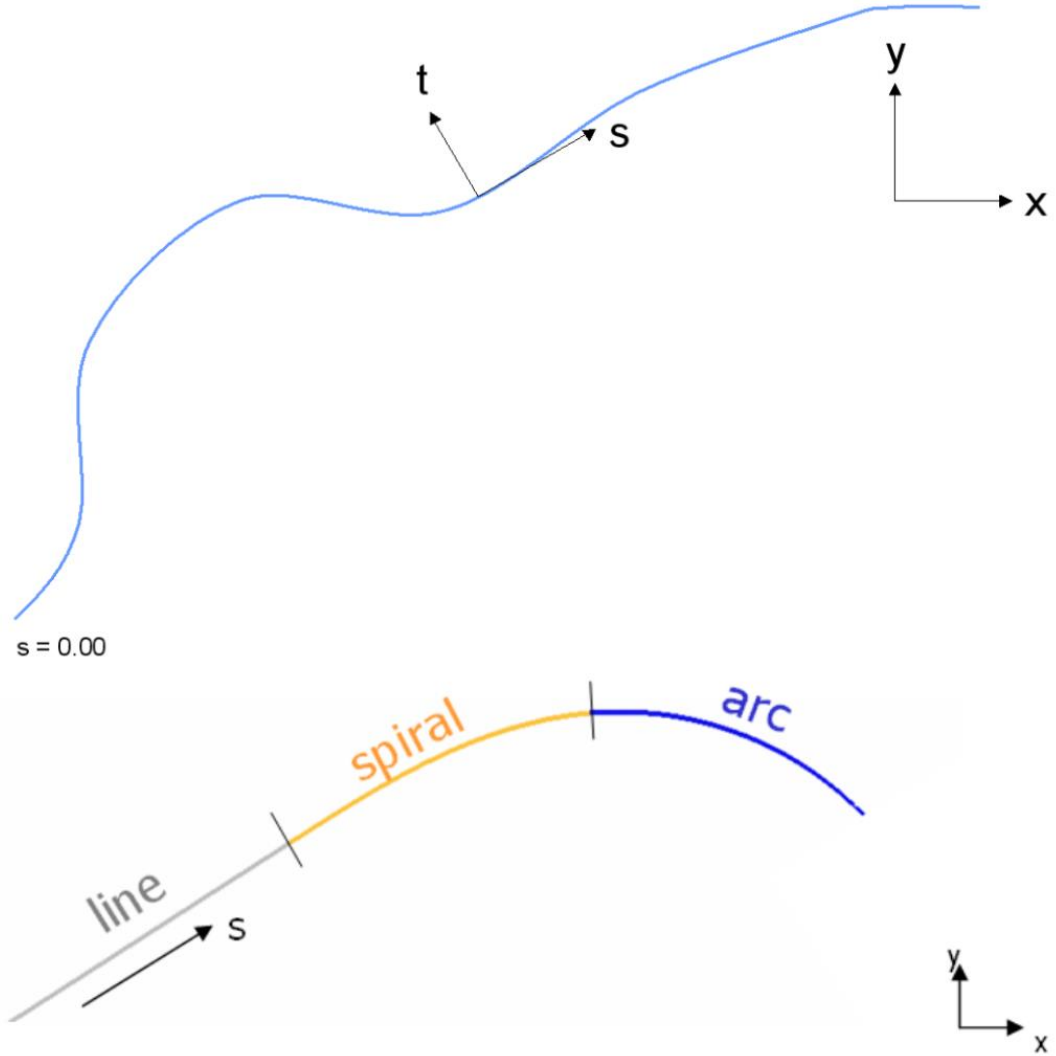
- Elements refer to an imaginary reference line



see OpenDRIVE Format Specification, Rev. 1.5

Make OpenDRIVE data GIS-able
Geometry basics

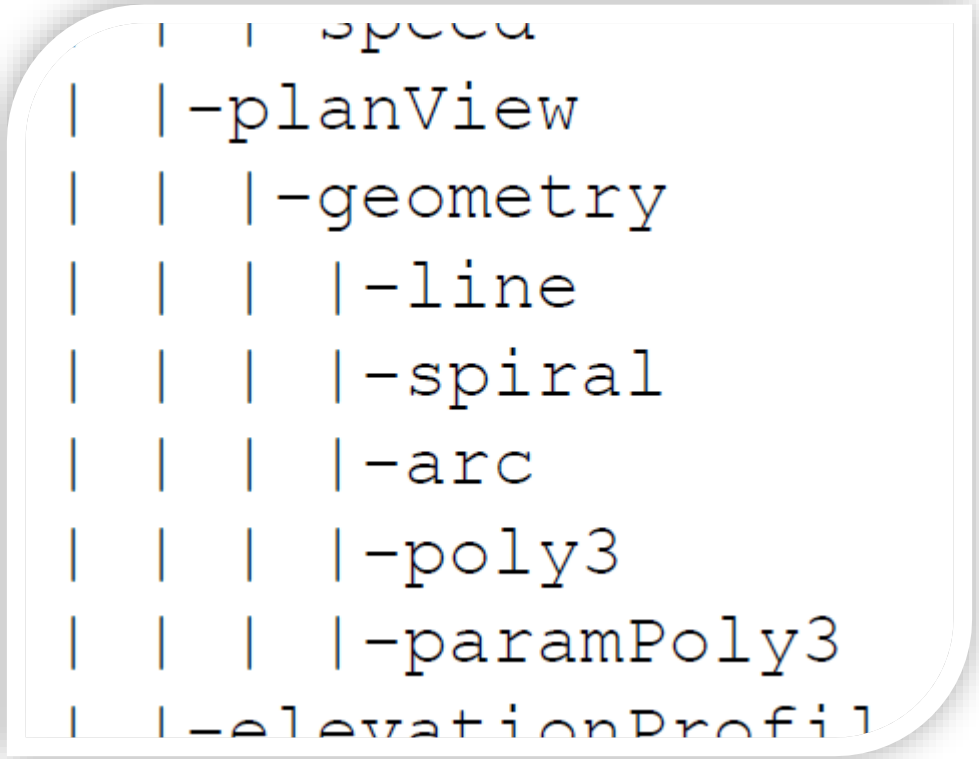
- Elements refer to an imaginary reference line
- Road topography (3D) and topology
  - continuous geometry definition



see OpenDRIVE Format Specification, Rev. 1.5

Make OpenDRIVE data GIS-able

Geometry basics: types



see OpenDRIVE Format Specification, Rev. 1.5

Make OpenDRIVE data GIS-able

Geometry basics: discrete anchor points

```
<planView>
  <geometry s="0.0" x="604944.1037"
            y="5792860.1272"
            hdg="3.5148"
            length="9.7589">
    <arc curvature="9.0884E-4"/>
  </geometry>
  <geometry s="9.7589" x="604935.03"
            y="5792856.5285"
            hdg="3.5237"
            length="12.0">
    <line/>
  </geometry>
</planView>
```

Make OpenDRIVE data GIS-able

Geometry basics: continuous geometry evolution

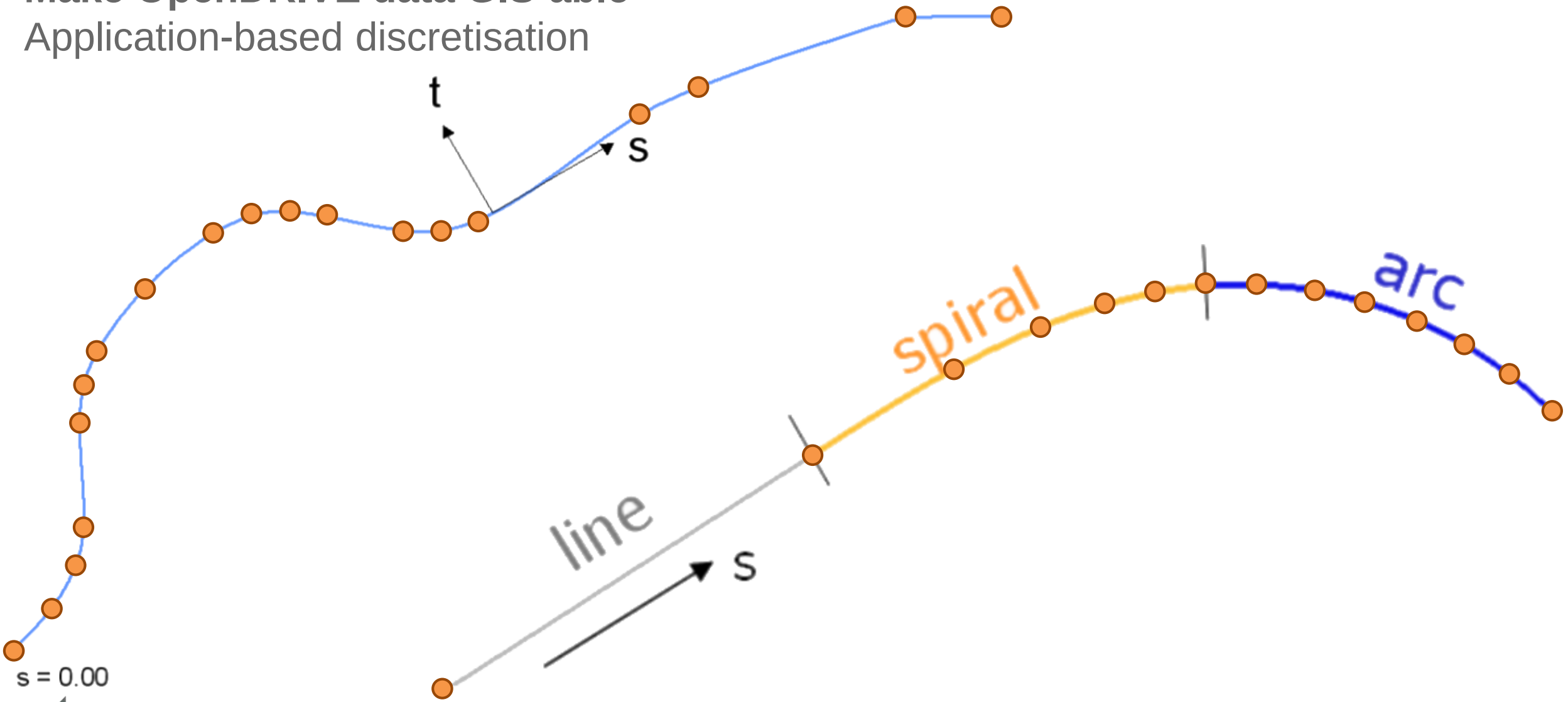
```
<planView>
  <geometry s="0.0" x="604944.1037"
            y="5792860.1272"
            hdg="3.5148"
            length="9.7589">
    <arc curvature="9.0884E-4"/>
  </geometry>
  <geometry s="9.7589" x="604935.03"
            y="5792856.5285"
            hdg="3.5237"
            length="12.0">
    <line/>
  </geometry>
</planView>
```

Make OpenDRIVE data GIS-able
Application-based discretisation (sampling)

“Everyone is doing this!”

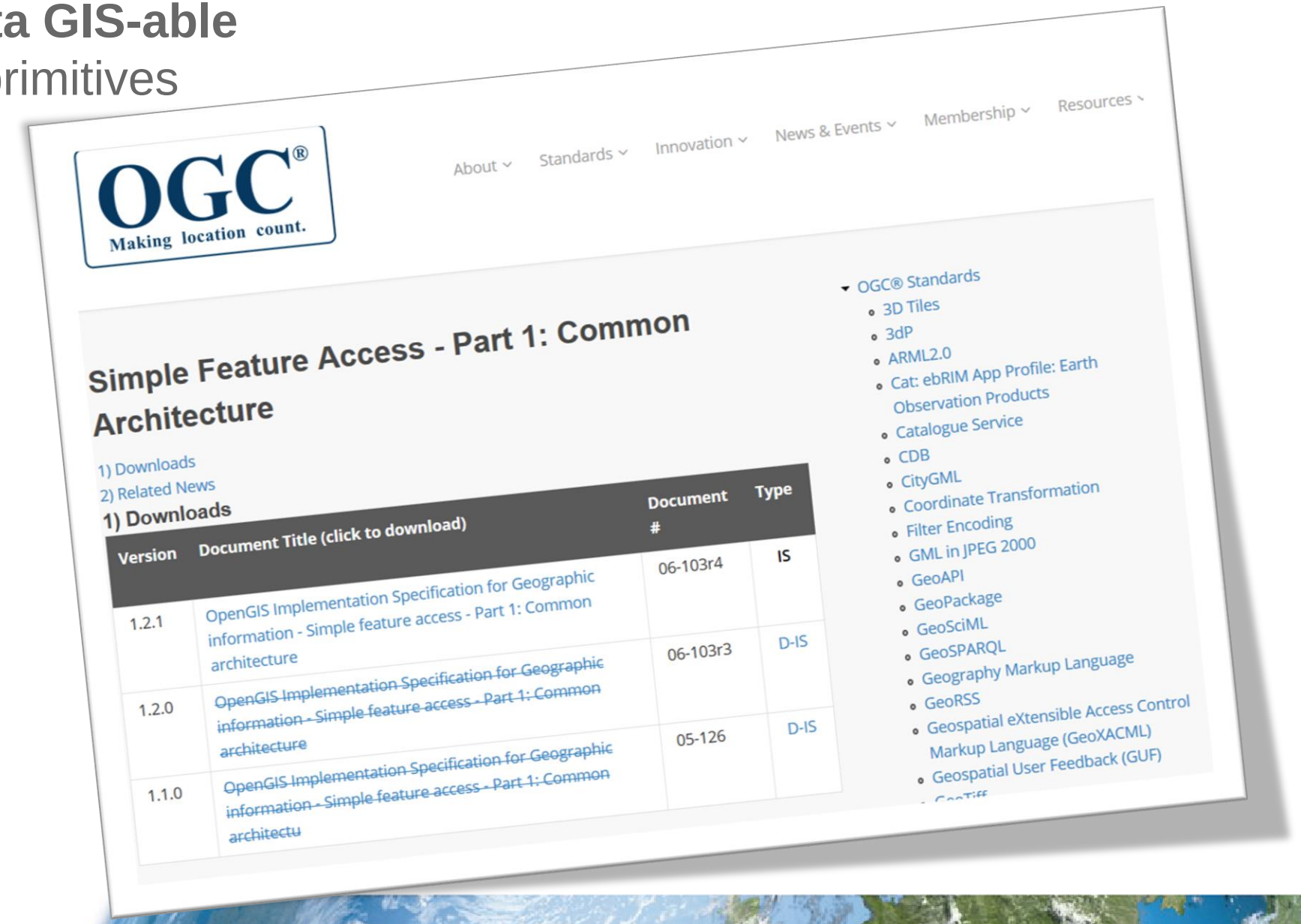
Make OpenDRIVE data GIS-able

Application-based discretisation

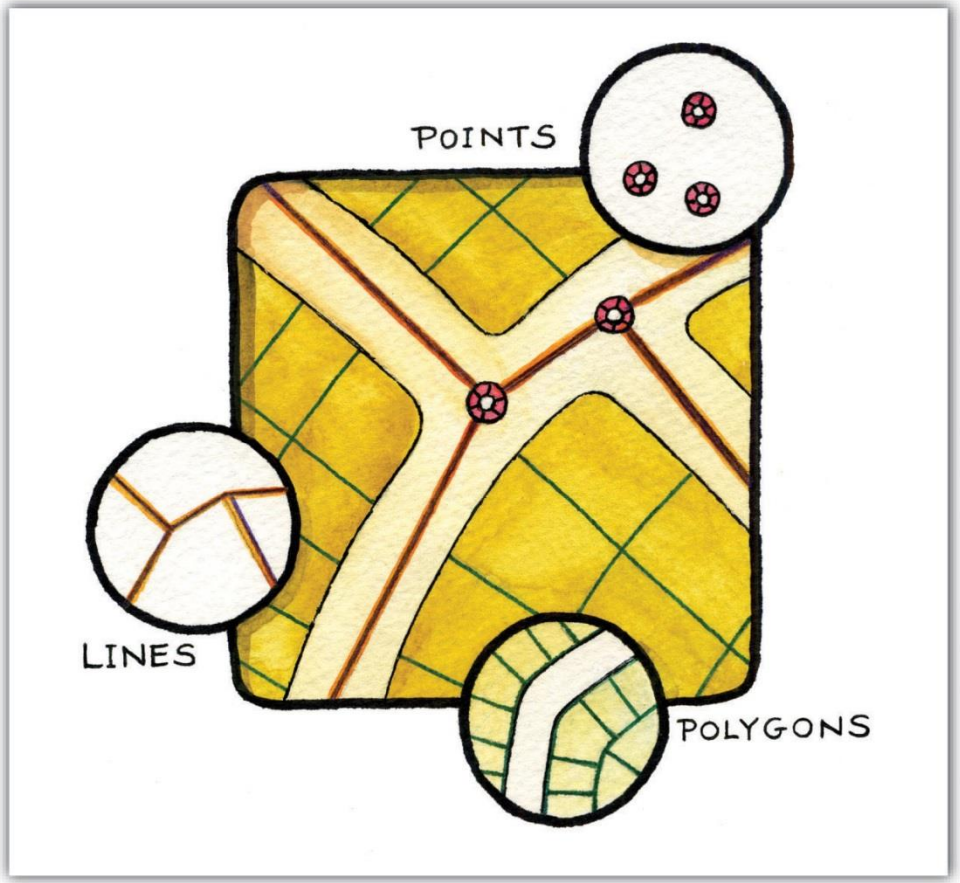


Make OpenDRIVE data GIS-able

OGC Simple Feature primitives

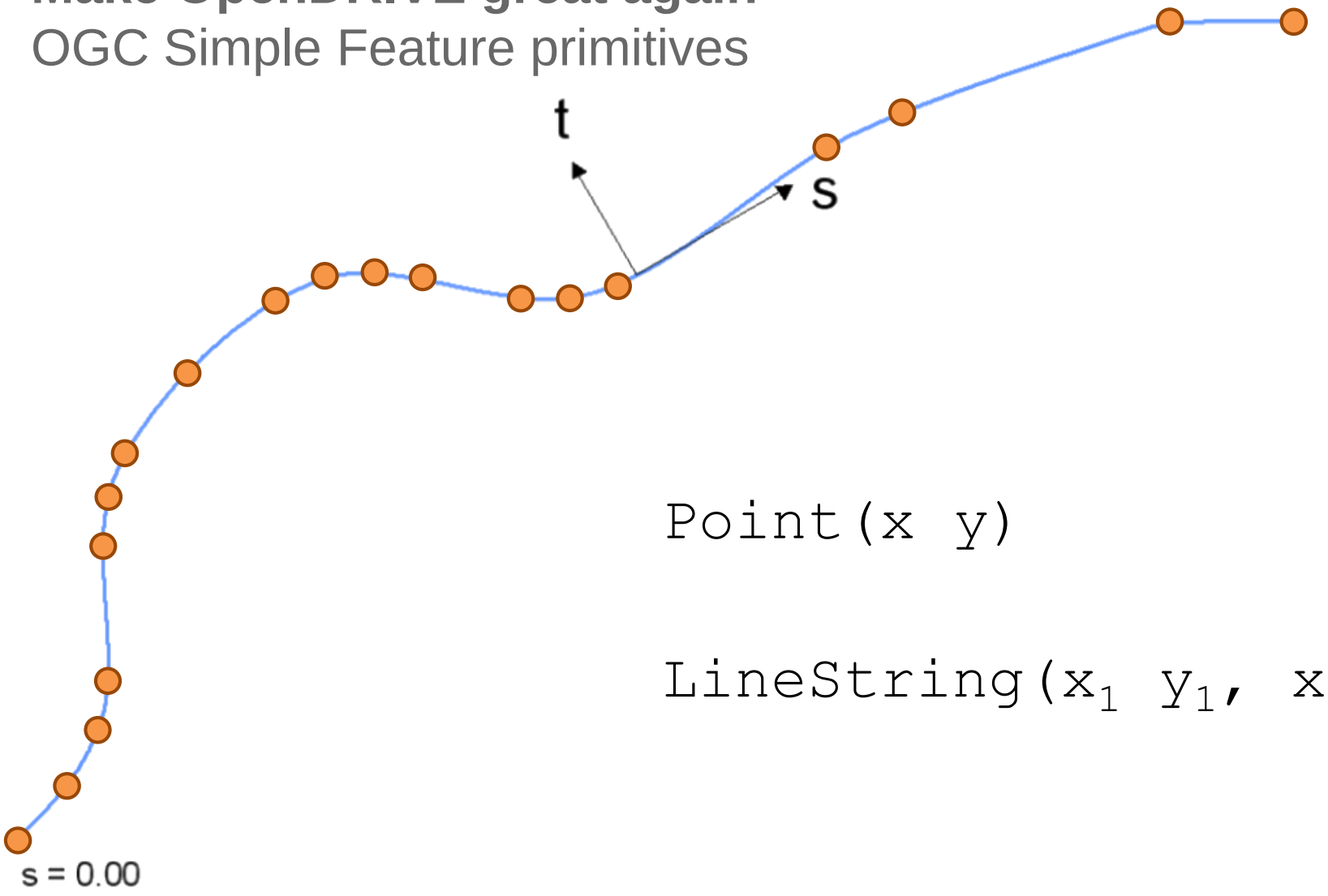


Make OpenDRIVE data GIS-able
OGC Simple Feature primitives



Make OpenDRIVE great again

OGC Simple Feature primitives



`Point(x y)`

`LineString(`x_1 y_1`,` x_2 y_2`, ...,` x_n y_n`)`

DLR

Make OpenDRIVE data GIS-able

Translation into Simple Feature model

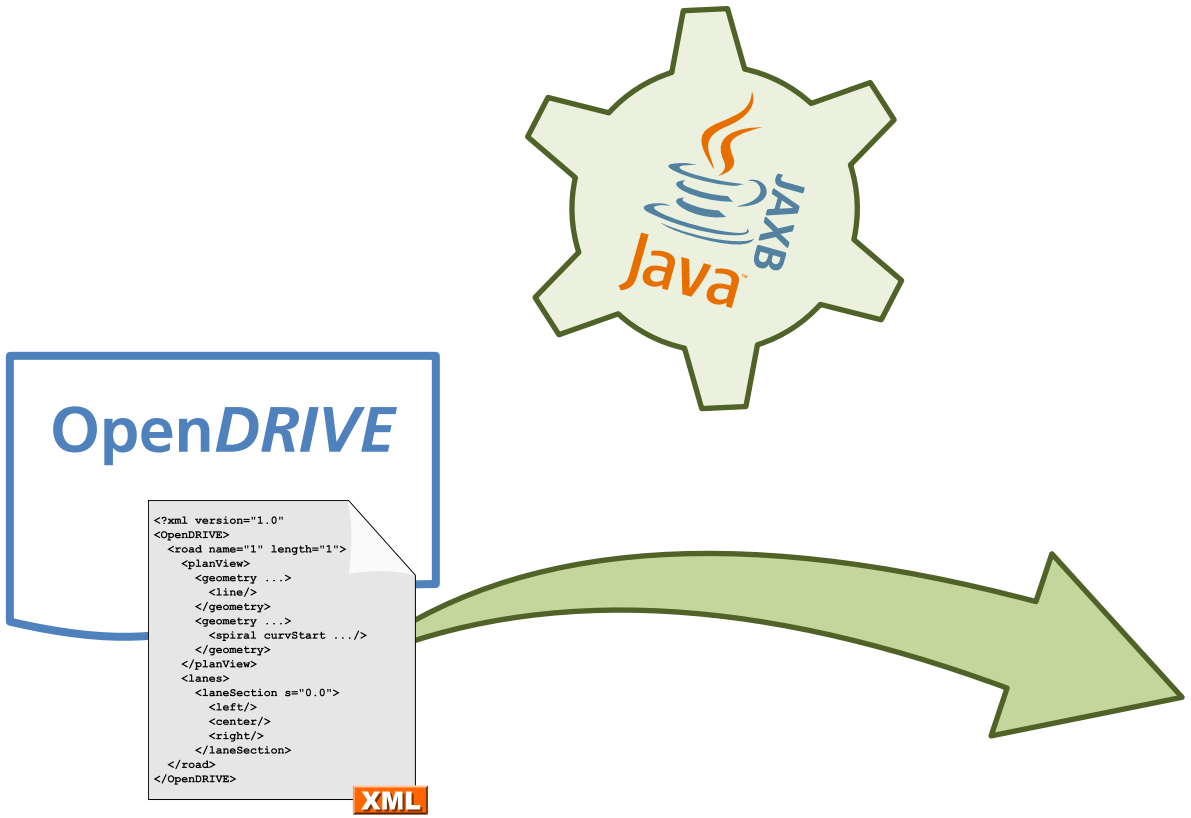
```
<planView>
  <geometry s="0.0" x="604944.1037"
            y="5792860.1272"
            hdg="3.5148"
            length="9.7589">
    <arc curvature="9.0884E-4"/>
  </geometry>
  <geometry s="9.7589" x="604935.03"
            y="5792856.5285"
            hdg="3.5237"
            length="12.0">
    <line/>
  </geometry>
</planView>
```

```
LineString(
   604944.1037 5792860.1272,
   604752.81 5792819.10, ...)

LineString(
  604935.03 5792856.5285,
  604754.39 5792810.73, ...)
```

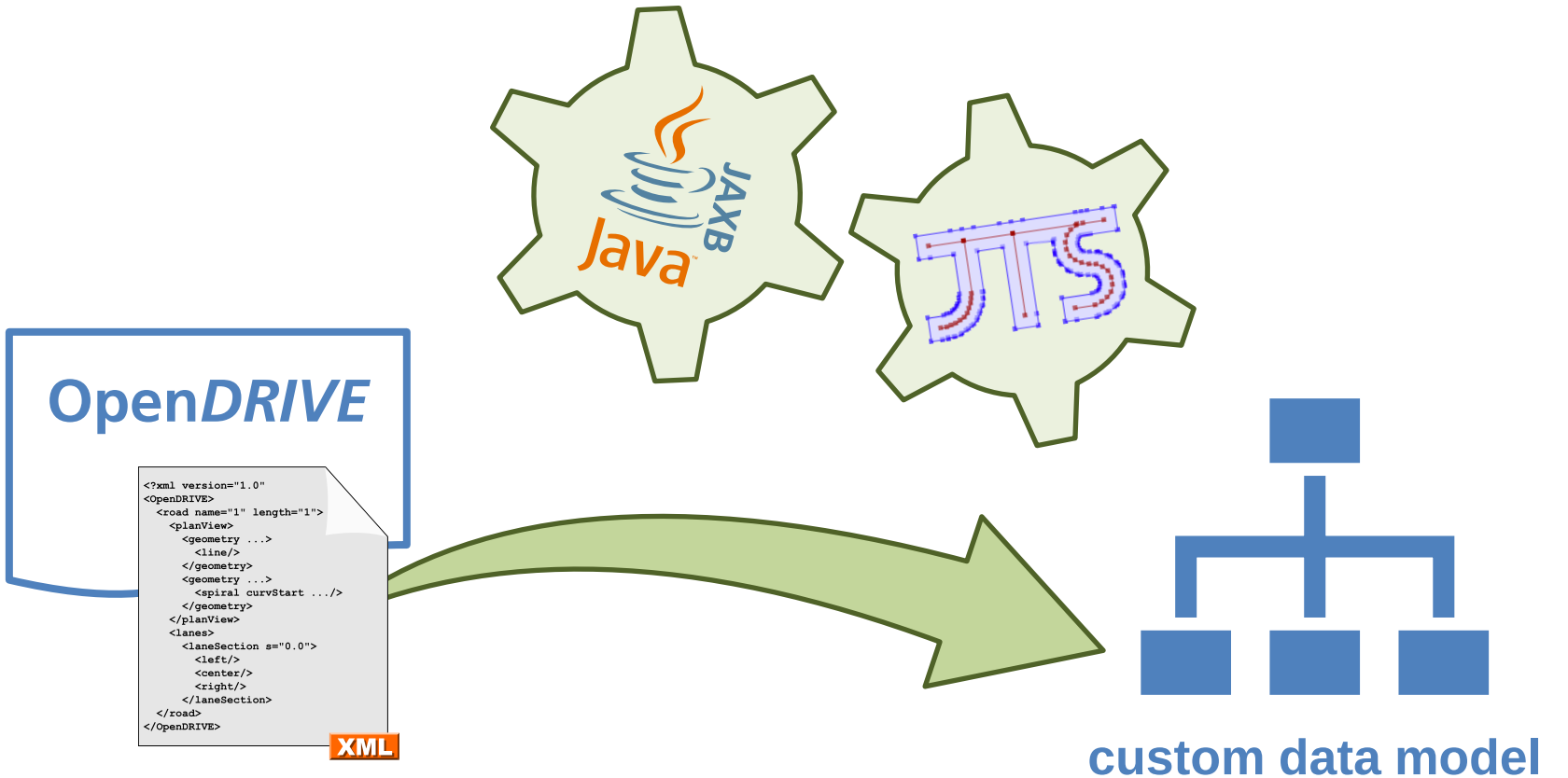
Make OpenDRIVE data GIS-able

Data binding to Java



Make OpenDRIVE data GIS-able

Geometry discretisation into Simple Feature model

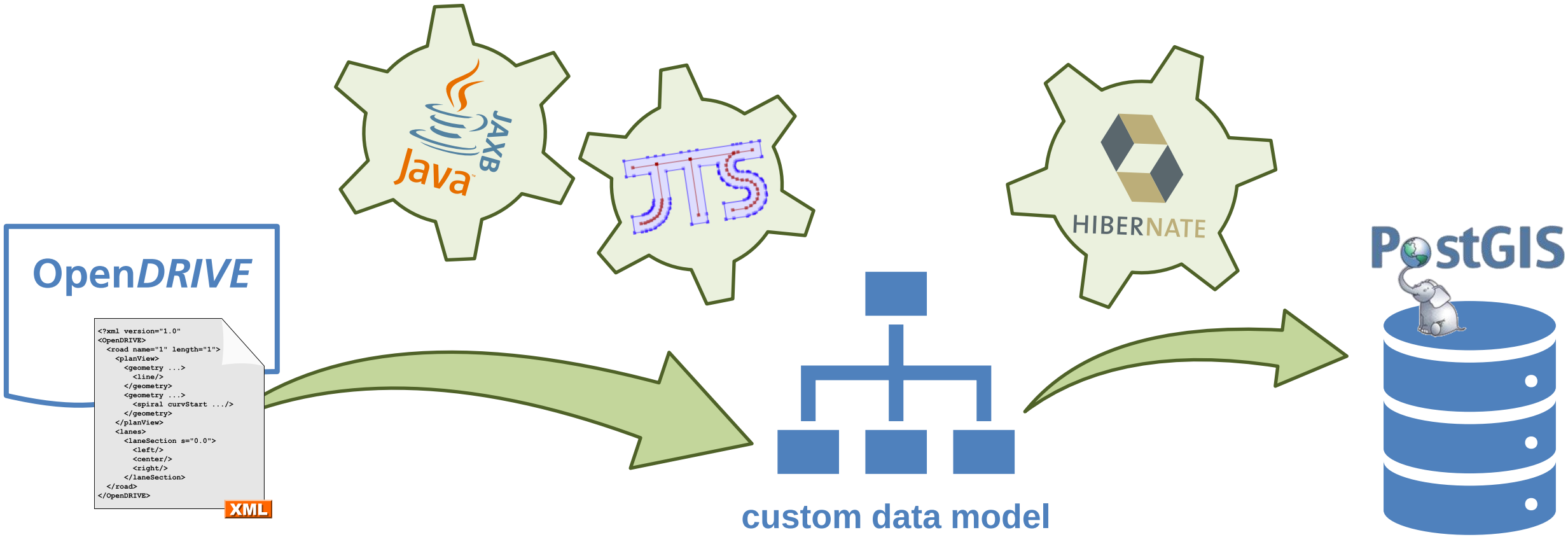


Solving “the problem” in three steps

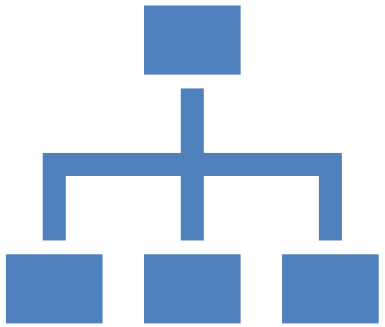
1. Make OpenDRIVE data GIS-able
2. Deploy GIS data in spatial database
3. Publish as RESTful web service

Deploy GIS data in spatial database

Persisting through JPA with Hibernate Spatial



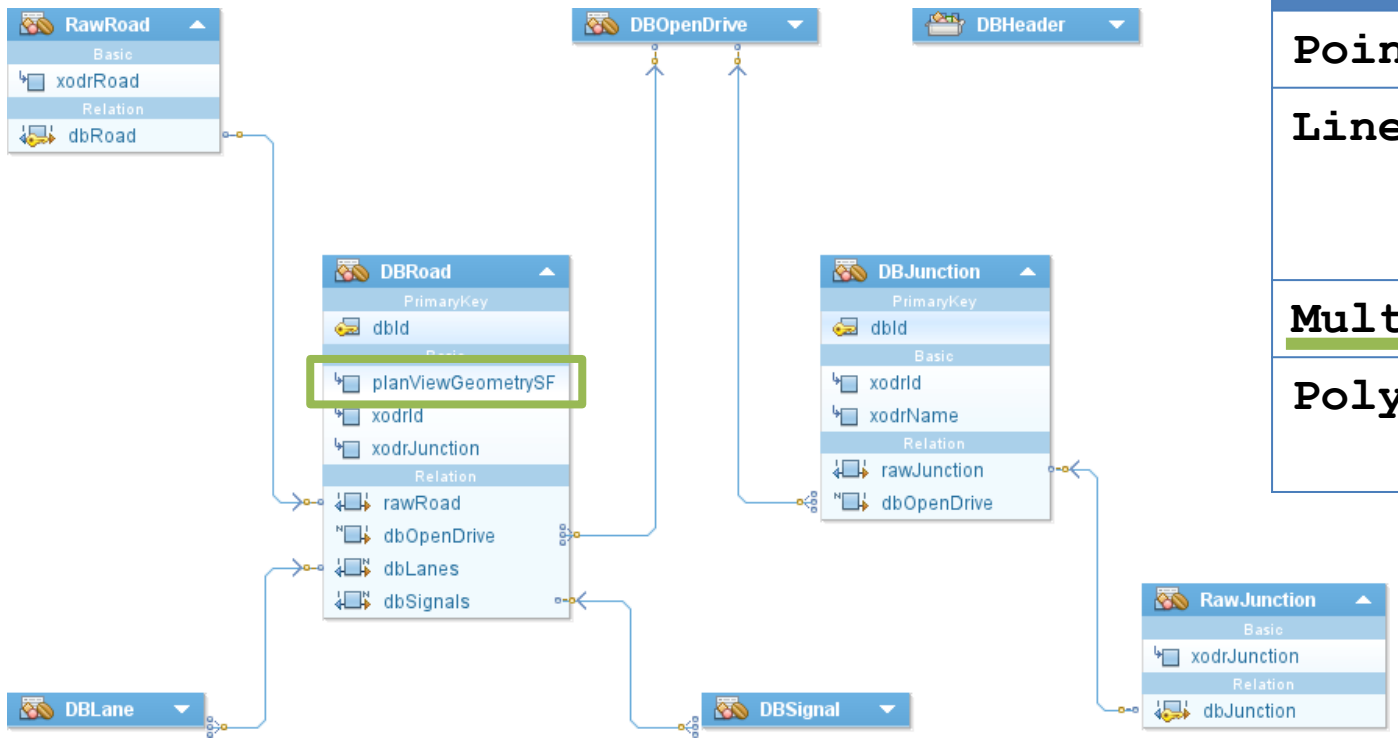
Deploy GIS data in spatial database
Custom data model



custom data model with Simple Features

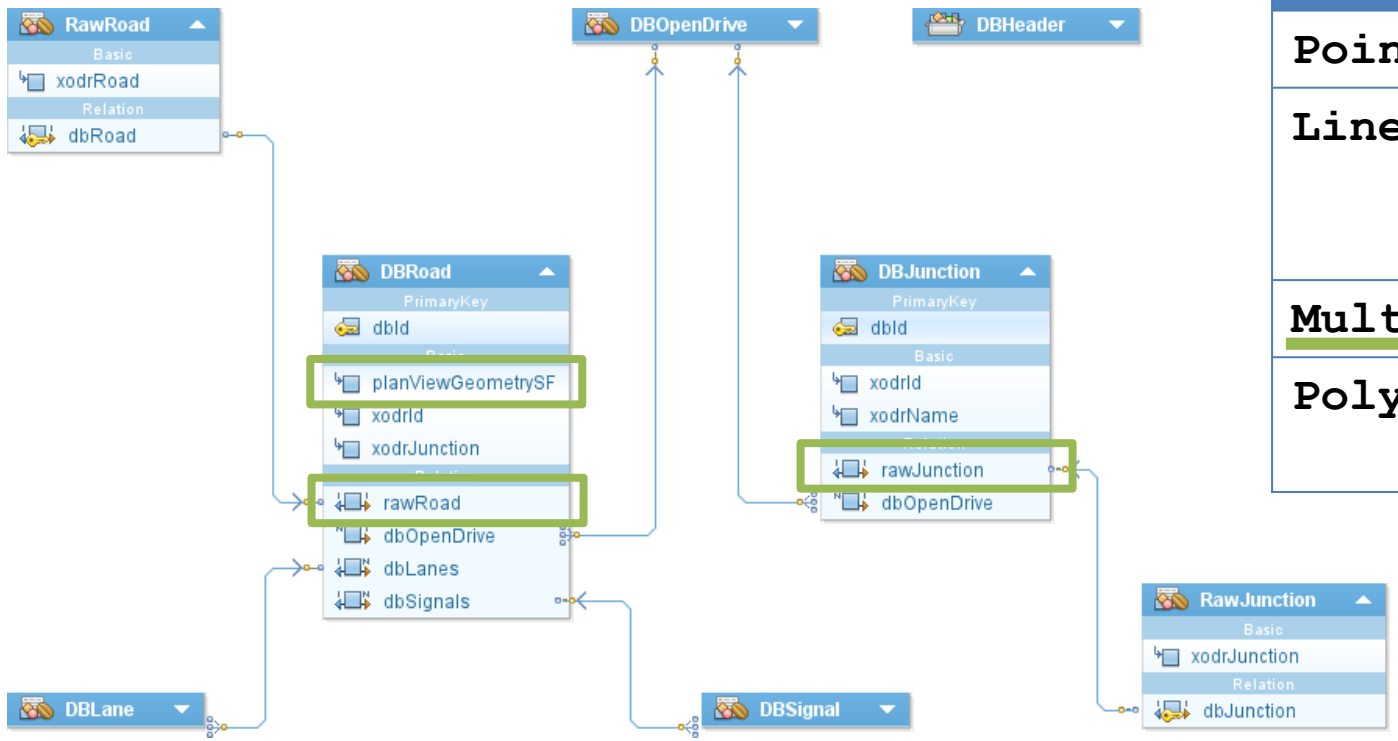
Simple Feature type	OpenDRIVE element
`Point`	`<signal>`
`LineString`	driving `<lane>` boundary, `<roadMark>`, linear `<object>` (e.g. guardrail)
`MultiLineString`	road reference line `<planView>`
`Polygon`	driving `<lane>`, `<parkingSpace>`

Deploy GIS data in spatial database
Custom data model



Simple Feature type	OpenDRIVE element
`Point`	`<signal>`
`LineString`	driving `<lane>` boundary, `<roadMark>`, linear `<object>` (e.g. guardrail)
`MultiLineString`	road reference line `<planView>`
`Polygon`	driving `<lane>`, `<parkingSpace>`

Deploy GIS data in spatial database
Custom data model



Simple Feature type	OpenDRIVE element
Point	<signal>
LineString	driving <lane> boundary, <roadMark>, linear <object> (e.g. guardrail)
MultiLineString	road reference line <planView>
Polygon	driving <lane>, <parkingSpace>

Additional raw XML elements “attached”

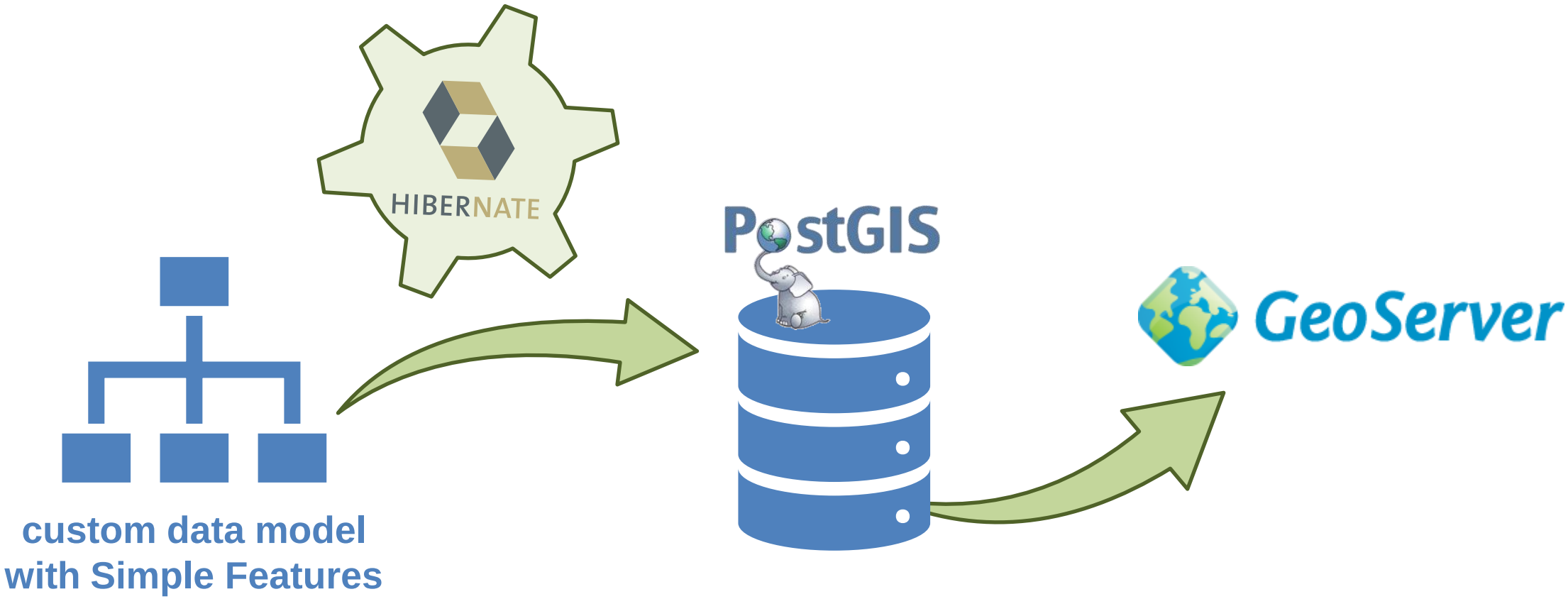
- <road>
- <junction>

Solving “the problem” in three steps

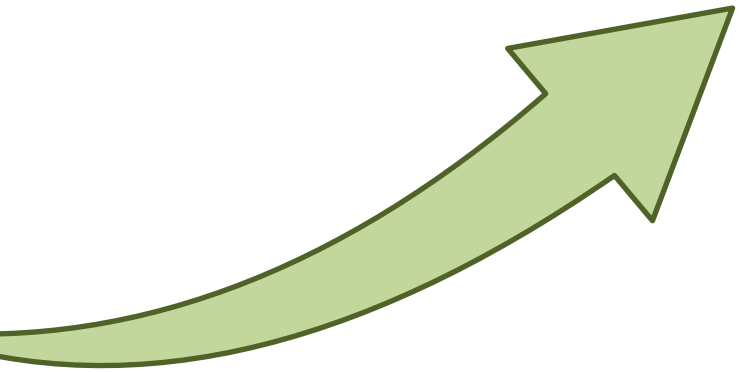
1. Make OpenDRIVE data GIS-able
2. Deploy GIS data in spatial database
3. Publish as RESTful web service

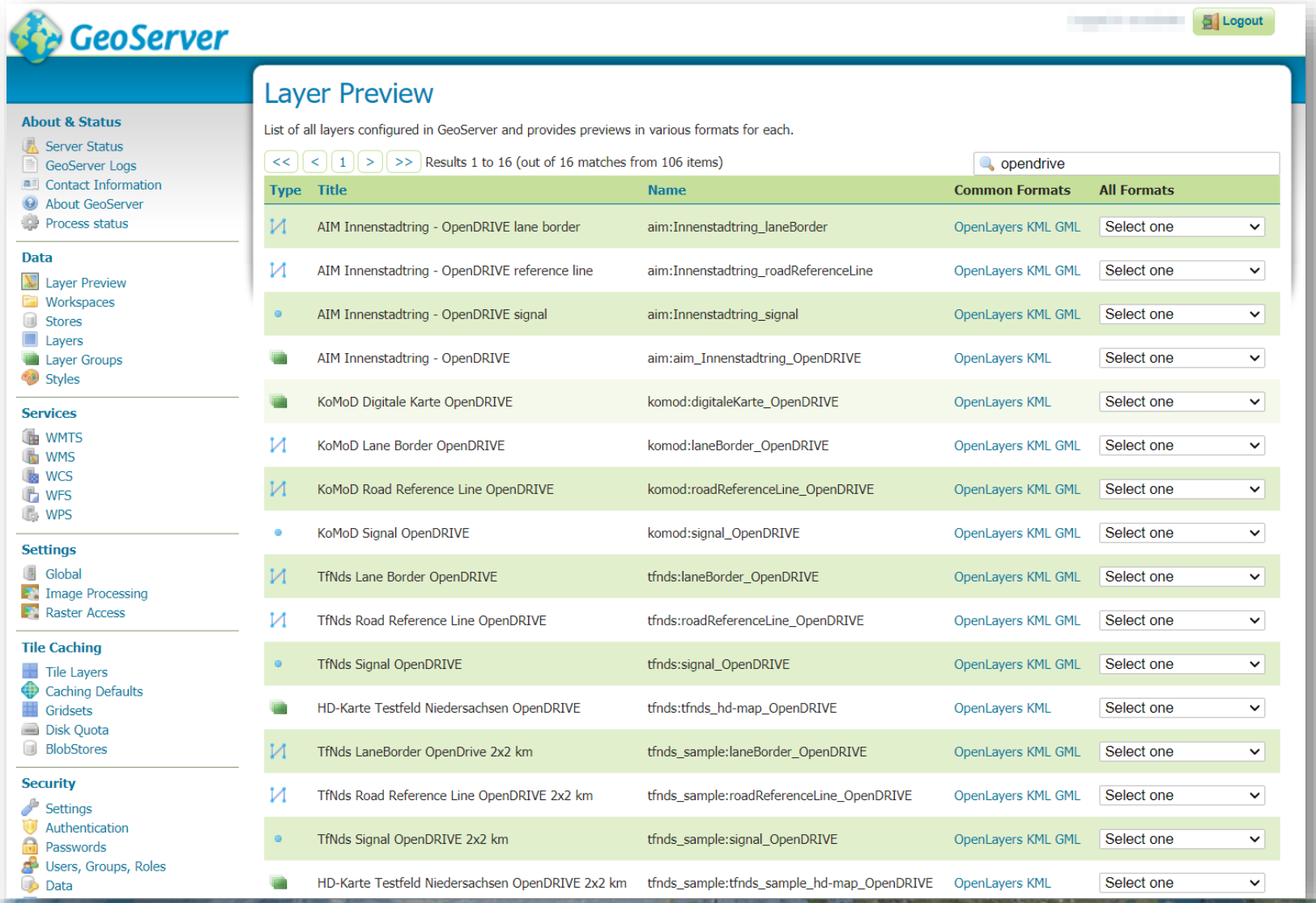
Publish as RESTful web service

Directly from database through GeoServer



Publish as RESTful web service





Publish as RESTful web service
The power of GeoServer

- Detailed **user management**
- Fine-grained **data security**/access policies
- OGC-standardised **REST API** (this “web thing”)
  - Web Map Service (WMS)
  - Web Feature Service (WFS)
  - Web Processing Service (WPS)
- Data output as image, KML, GML, GeoJSON, CSV, Shapefile, etc.
- Easy **snippet extraction** through custom **extension**
- Benefiting **spatial indices** on data → “fast like hell”
- Scalability
- Integration into most GIS tools
- Bla, bla, bla …

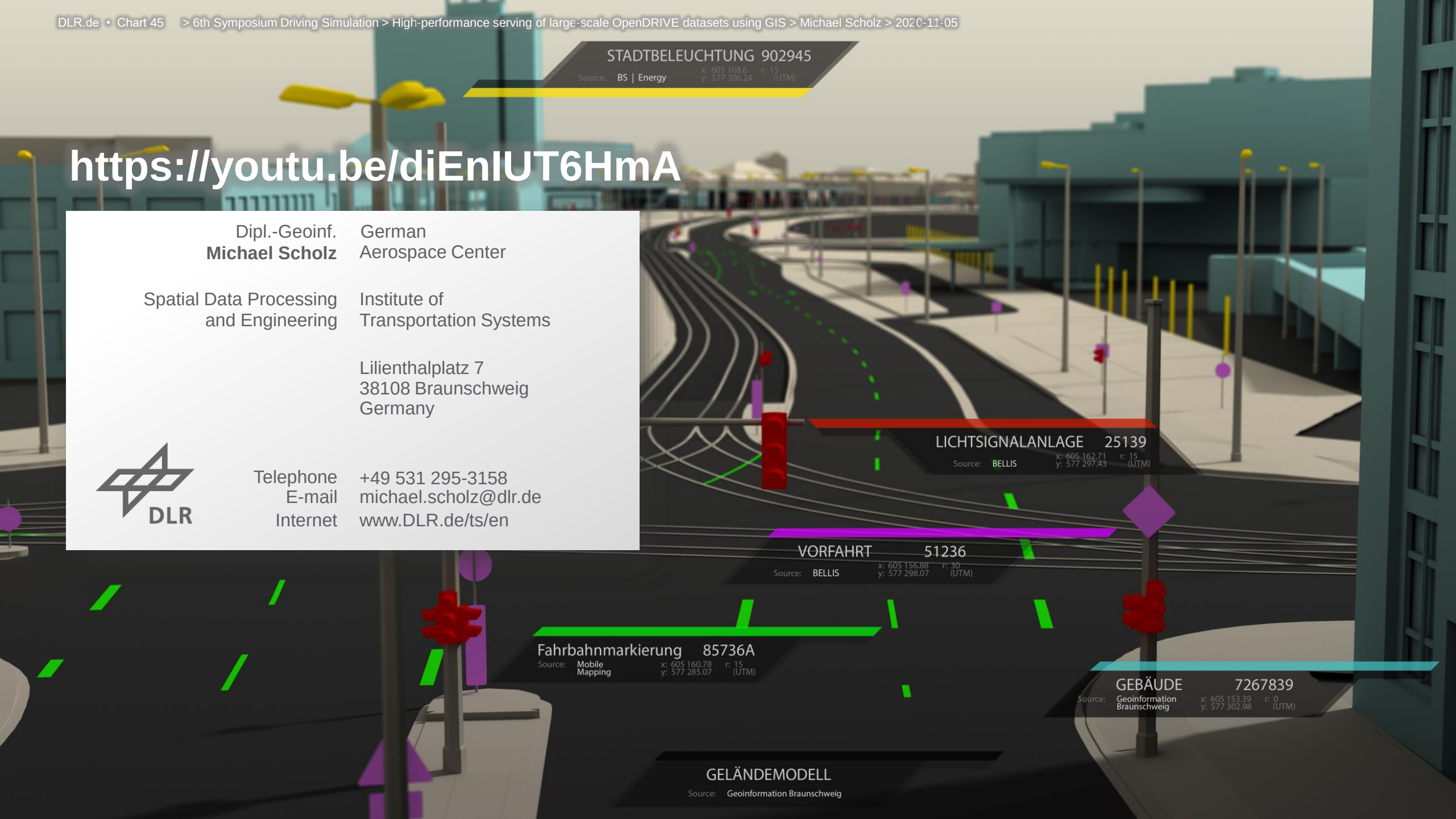
Live demo

OpenDRIVE through GeoServer
OpenDRIVE in QGIS
OpenDRIVE subset/snippet extraction queries

Conclusion: “Don’t re-invent the wheel”

- Geometry discretization should be based on OGC Simple Features
- Benefit from well-established tools in GIS domain:
  - Free/open frameworks for Java, C++, Python, … and web development
  - Super-easy ad hoc combination with arbitrary geo-data
  - Direct conversion into 100+ other formats: KML, GML, GeoJSON, CSV, Shapefile, SQLite, XLSX, …
    → GDAL: “One library to rule them all”
  - Standardized web service interfaces already implemented (OGC WMS, WFS, …)

This ecosystem enables fast, large-scale serving of OpenDRIVE



https://youtu.be/diEnIUT6HmA

Dipl.-Geoinf. **Michael Scholz**

German Aerospace Center

Spatial Data Processing and Engineering

Institute of Transportation Systems

Lilienthalplatz 7
38108 Braunschweig
Germany

DLR

Telephone +49 531 295-3158
E-mail michael.scholz@dlr.de
Internet www.DLR.de/ts/en