railML.org

Highlights railML 3.2 Common

Thomas Nygreen, common coordinator, railML.org

Agenda

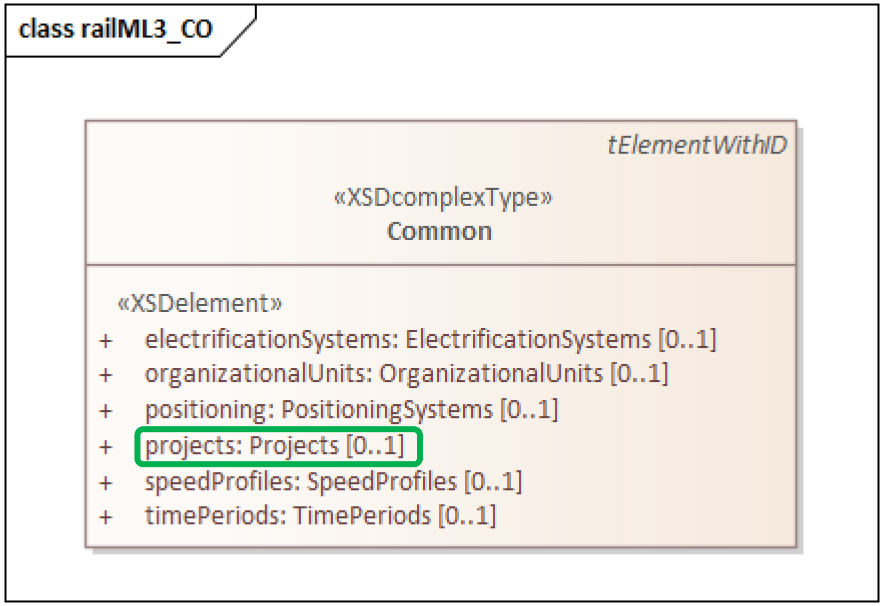
1. New feature: Metadata for project descriptions
2. Handling changes between minor versions
3. Merging railML 2 and railML 3 code lists
4. **Facilitating extensions for railML 3:**
   <any> vs. polymorphism – subtyping and xsi:type

1: Metadata for project descriptions

<project>

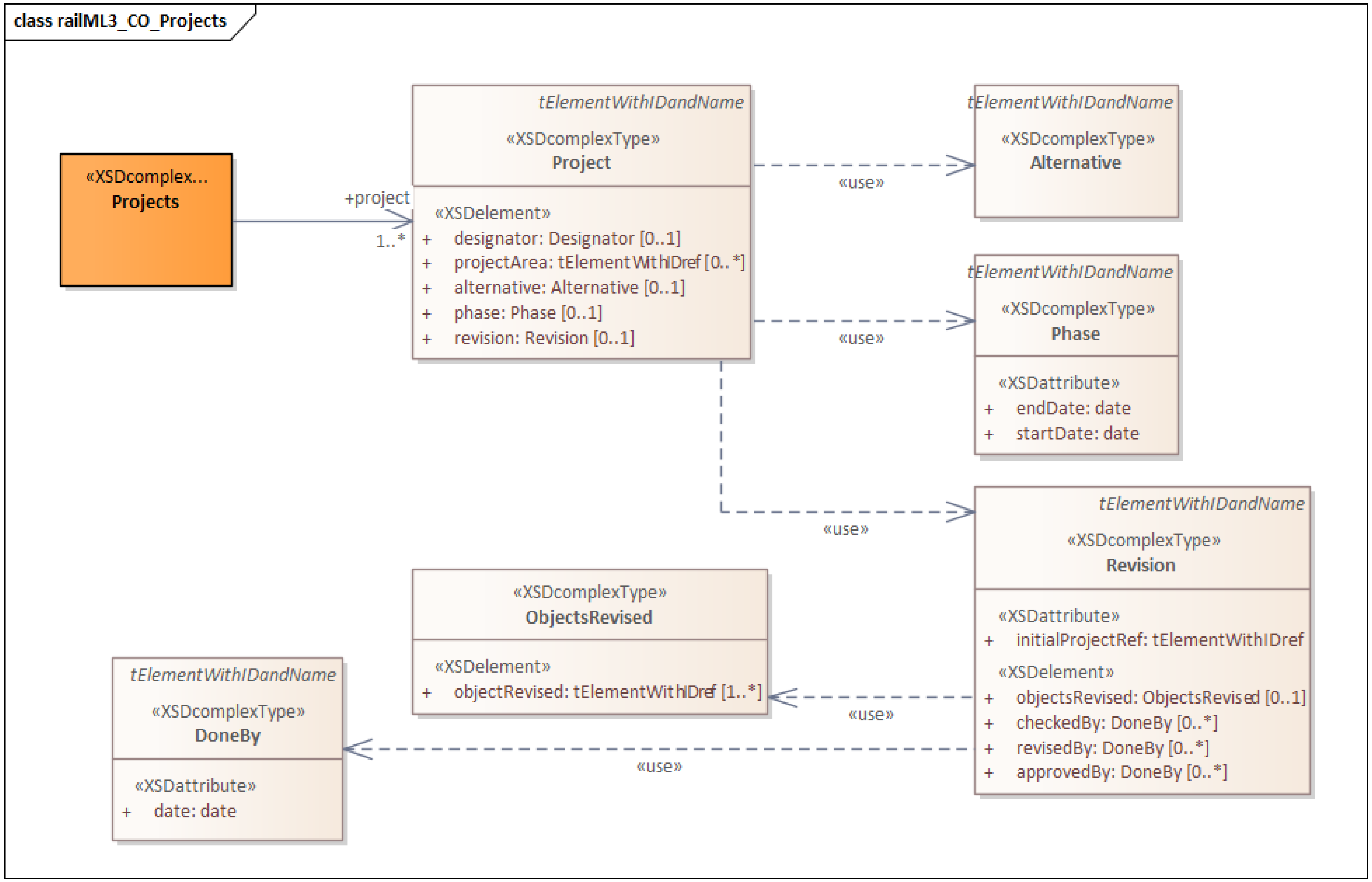
Describes the purpose of the infrastructure objects which are part of a model. One model can have multiple projects.

Implemented similarly in railML 3.2 as in railML 2.5.


class railML3_CO
tElementWithID
«XSDcomplexType»
Common
«XSDelement»
+ electrificationSystems: ElectrificationSystems [0..1]
+ organizationalUnits: OrganizationalUnits [0..1]
+ positioning: PositioningSystems [0..1]
+ projects: Projects [0..1]
+ speedProfiles: SpeedProfiles [0..1]
+ timePeriods: TimePeriods [0..1]

railML.org

1: Metadata for project descriptions (continued)



2: Handling changes between minor versions

As previously announced:

Changes may lead to elements or attributes being deprecated. Elements and attributes that are deprecated in one minor version will be removed in the following minor version. Compatibility is only guaranteed between one minor version and the next.

Concerning other changes, we will try to follow the same strategy of compatibility between two consecutive minor versions. Specifically:

Changing an attribute or element from required to optional:
railML 3.1: required
railML 3.2: required, with a notice that the «required» status is deprecated
railML 3.3: optional

3: Merging railML 2 and railML 3 code lists

The code lists should be generic and «universal»

- Will be shared between railML 2 and railML 3
- Current (unintentional) differences will be merged
- Wiki documentation will be updated
- Shared code lists will have a separate repository

Code lists will still be updated continuously, independent of railML version development.

4: Facilitating extensions for railML 3

Extending railML – Why?

railML is constantly being improved to meet the railway data exchange needs of the international railway sector

However, some needs are:

- Company-specific
- Country-specific
- Application-specific
- Experimental

→Using railML as a base and add custom extensions

All extensions should be presented in the Forum. Maybe a best practise already exists?

railML Forum

The present: <xs:any> and <xs:anyAttribute>

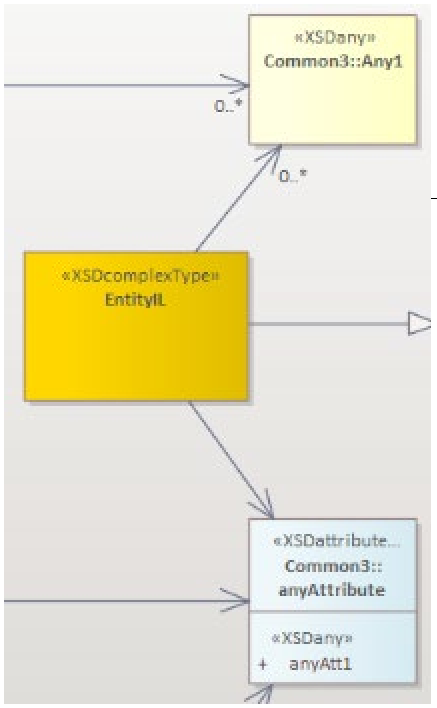
railML currently uses <xs:any> and <xs:anyAttribute> to facilitate custom elements or attributes

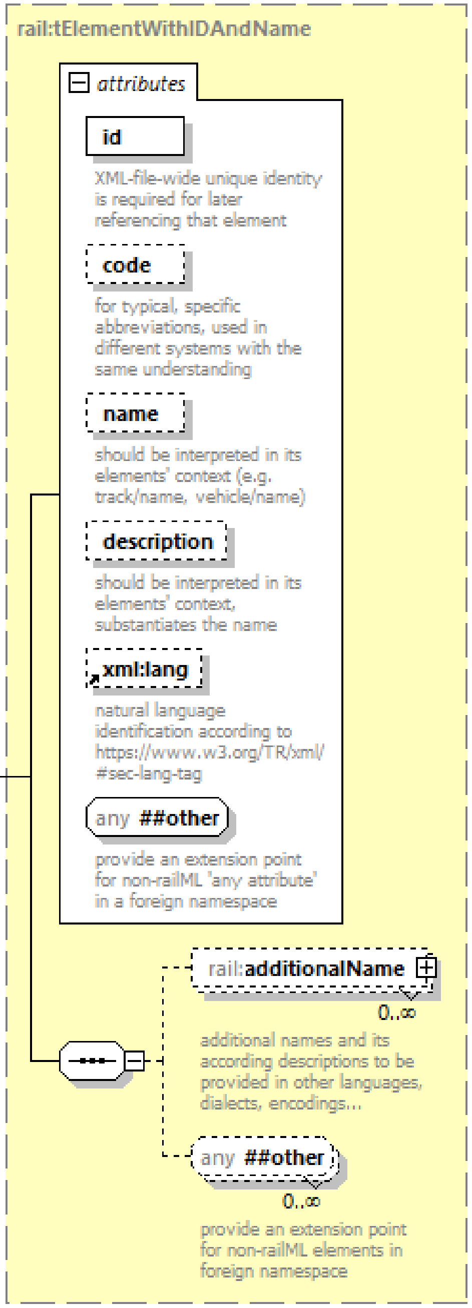
+ Flexible: Allows placing new elements and attributes wherever they are needed

– New elements and attributes cannot be places on specific elements

– Creates ambiguity (=invalid schema) when reusing types in other namespaces

– Code generation tools don't understand them



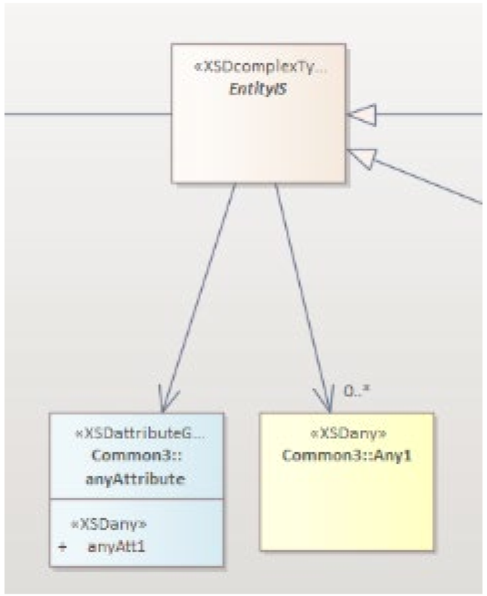

rail:tElementWithIDAndName
attributes
id
XML-file-wide unique identity is required for later referencing that element
code
for typical, specific abbreviations, used in different systems with the same understanding
name
should be interpreted in its elements' context (e.g. track/name, vehicle/name)
description
should be interpreted in its elements' context, substantiates the name
xml:lang
natural language identification according to https://www.w3.org/TR/xml/#sec-lang-tag
any ##other
provide an extension point for non-railML 'any attribute' in a foreign namespace
rail:additionalName
0..∞
additional names and its according descriptions to be provided in other languages, dialects, encodings...
any ##other
0..∞
provide an extension point for non-railML elements in foreign namespace

Example using <xs:anyAttribute>

For some reason, I need to know if operational points have a fire station

So, I make an extension XML Schema with a new **global** attribute:

```
<xs:attribute name="hasFireStation" type="xs:boolean"/>
```



Then, I can do what I need:

```
<operationalPoints>
  <operationalPoint id="op1" ext:hasFireStation="true" />
  <operationalPoint id="op2" ext:hasFireStation="false" />
</operationalPoints>
```

railML.org

Example using <xs:anyAttribute> (continued)

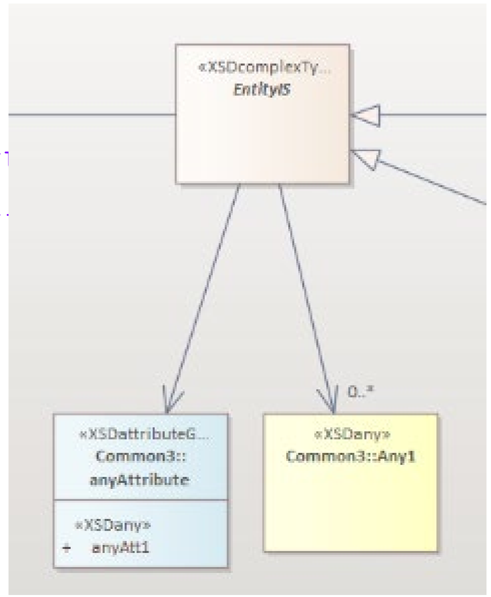
I can do what I need:

```
<operationalPoints>
 <operationalPoint id="op1" ext:hasFireStation="tr
 <operationalPoint id="op2" ext:hasFireStation="fa
</operationalPoints>
```

But I can also do strange things:

```
<derailerIS id="dr1" ext:hasFireStation="true" />
<route id="r1" ext:hasFireStation="true" />
```

→Computers get confused



Alternative: subtype polymorphism with xsi:type

XML Schema allows extending types from another namespace

```
<xs:complexType name="OperationalPoint">
  <xs:complexContent>
    <xs:extension base="rail3:OperationalPoint">
      <xs:attribute name="hasFireStation" type="xs:boolean" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

And XML allows subtyping with xsi:type

```
<operationalPoints>
  <operationalPoint xsi:type="ext:OperationalPoint"
                    id="op1" ext:hasFireStation="true" />
  <operationalPoint xsi:type="ext:OperationalPoint"
                    id="op2" ext:hasFireStation="false" />
</operationalPoints>
```

Pros and cons of subtyping

I can no longer do strange things:

~~`<derailerIS xsi:type="ext:OperationalPoint" ...`~~
~~`<route xsi:type="ext:OperationalPoint" ...`~~

Computers understand what goes where

- Code generation works!

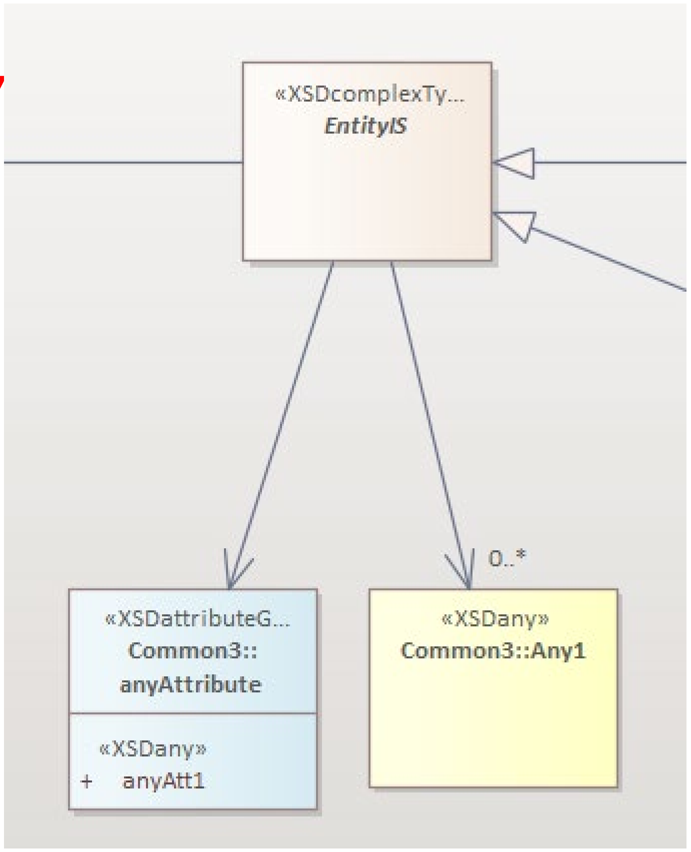
BUT:

- Extending with new elements does not work if the original type has an <any> extension point
- You cannot combine multiple subtype extensions on the same element with xsi:type – must be done in the extending schema

<any> breaks subtyping

```
<xs:complexType name="NewEntityType">
  <xs:complexContent>
    <xs:extension base="rail3:EntityIS">
      <xs:sequence>
        <xs:element name="newElement" ty
      </xs:sequence>
      <xs:attribute name="newAttribute"
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

INVALID



railML.org

Less flexible combination of extensions

$$\begin{tikzcd}
 & \boxed{\text{rail3:SignalIS}} & \\
\boxed{\text{rail3:SignalExtA}} \arrow[ur] & & \boxed{\text{rail3:SignalExtB}} \arrow[ul]
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{\text{rail3:SignalIS}} \\
\boxed{\text{rail3:SignalExtA}} \arrow[u] \\
\boxed{\text{rail3:SignalExtB}} \arrow[u]
\end{tikzcd}$$

<signalIS xsi:type="..."

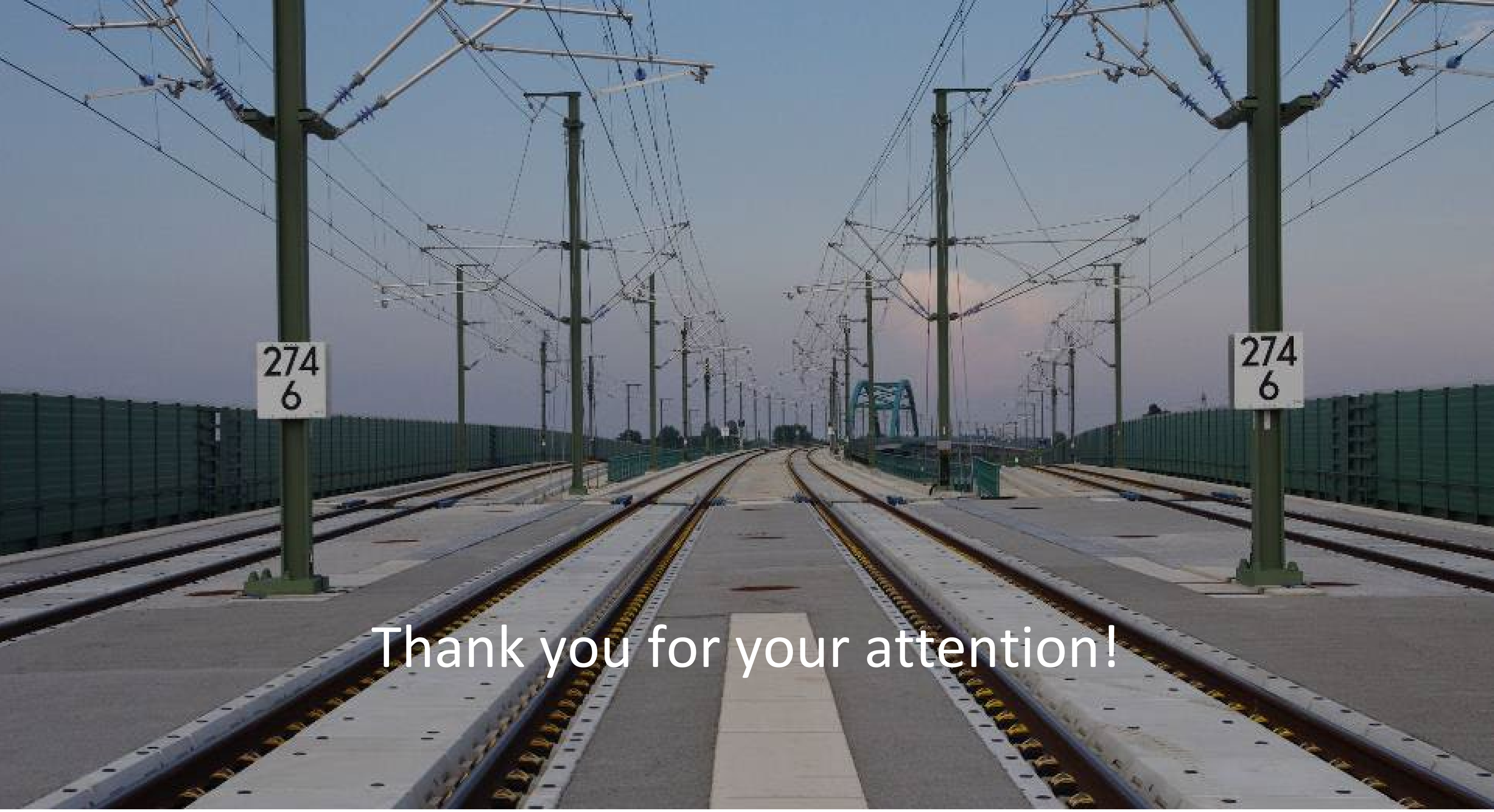
To xsi:type or not to xsi:type?

Should we keep <any> and <anyAttribute>, or move to xsi:type?

Our current recommendation is to deprecate and then remove <any>, but leave <anyAttribute>.

Please put your feedback in the forum:
https://www.railml.org/forum/index.php?t=msg&goto=2088

or by email to coord@common.railml.org



Thank you for your attention!

Jernbane-
direktoratet

www.railml.org

Thomas Nygreen, Norwegian Railway Directorate

Common coordinator, railML.org

coord@common.railml.org

+47 47 47 50 88