railML.org

Change policy and other Common news

Thomas Nygreen, common coordinator, railML.org

Agenda

1. Expanding the policy on changes between versions of railML 3
2. Other news
   - Substituting some `xs:sequences` with `xs:all` or `xs:choice`.
   - Generalise <infrastructureState> to use with the other subschemas

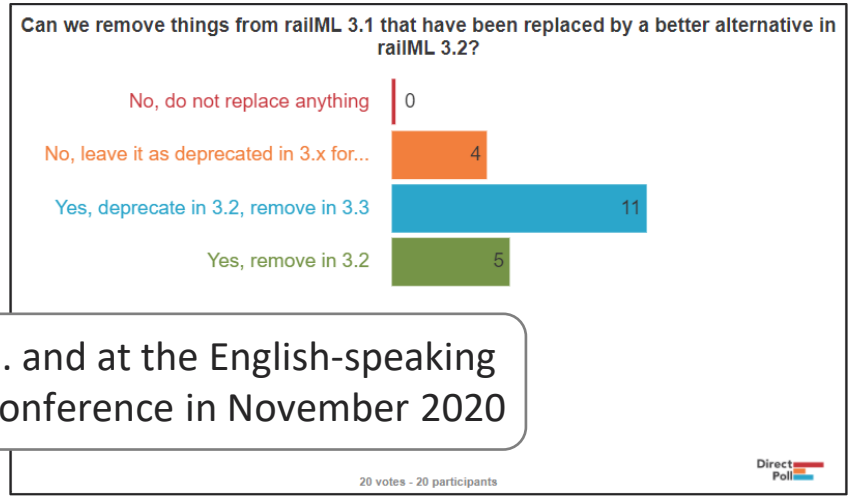
Part 1: Change/deprecation policy

The current change/deprecation policy for railML 3

We [asked you in the forum in 2020](), and decided in January 2021 that:

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.

[Handling minor changes within versions - railML.org]()



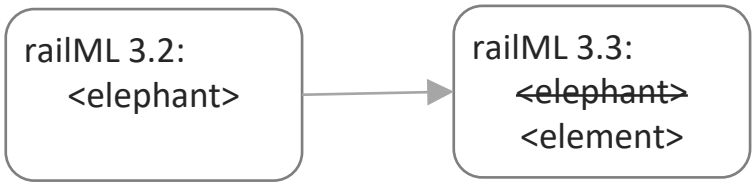
... and at the English-speaking conference in November 2020

And we checked at the German-speaking conference in April 2021

The current change/deprecation policy for railML 3

- Added more flexibility to make changes in existing modelling than in railML 2
- Works for removing something that has been superceded
- Does not cover cases where old and new modelling cannot coexist
- Increases complexity when interpreting a railML file

Draft for new policy



Changes from one minor version to the next will be allowed without a deprecation phase.

This allows changes that cannot have a deprecation phase, such as:

- Renaming elements, attributes or enumeration values
- Changing the minOccurs or maxOccurs of an element
- Changing the use of an attribute (optional or mandatory)
- Changing between xs:sequence, xs:choice and xs:all

Remodelling may also be done without including both the old and new implementation in the new version, reducing the complexity.

Removals that are not replaced by something new may still have a deprecation phase.

Part 2: Other news

Substituting some `xs:sequences` with `xs:all` or `xs:choice`

- Some groups of elements that were intended to be modelled as a choice, are for technical reasons defined as sequences.
- Now that we are not using xs:any extensions, we can also switch some sequences for xs:all.
- Feedback from the community was that we should change away from sequences where appropriate.
- Work in progress.

xs:sequence

- Individual multiplicity
- Same order as in the XSD
- Supports xs:any

xs:all

- Multiplicity only 0..1 or 1..1
- Any order
- Does not support xs:any (no longer relevant)

xs:choice

- Select one of the possible child elements
- Can be repeatable or not

Generic state for all subschemas

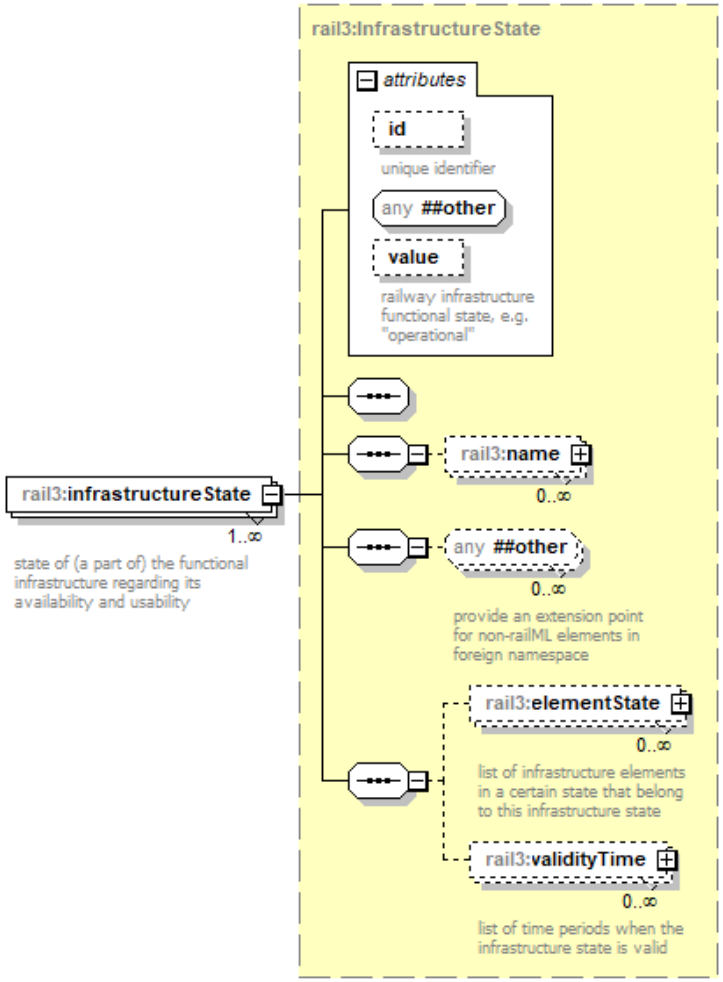
Currently we have <infrastructureStates> to describe if an infrastructure element is operational, planned, disabled etc.

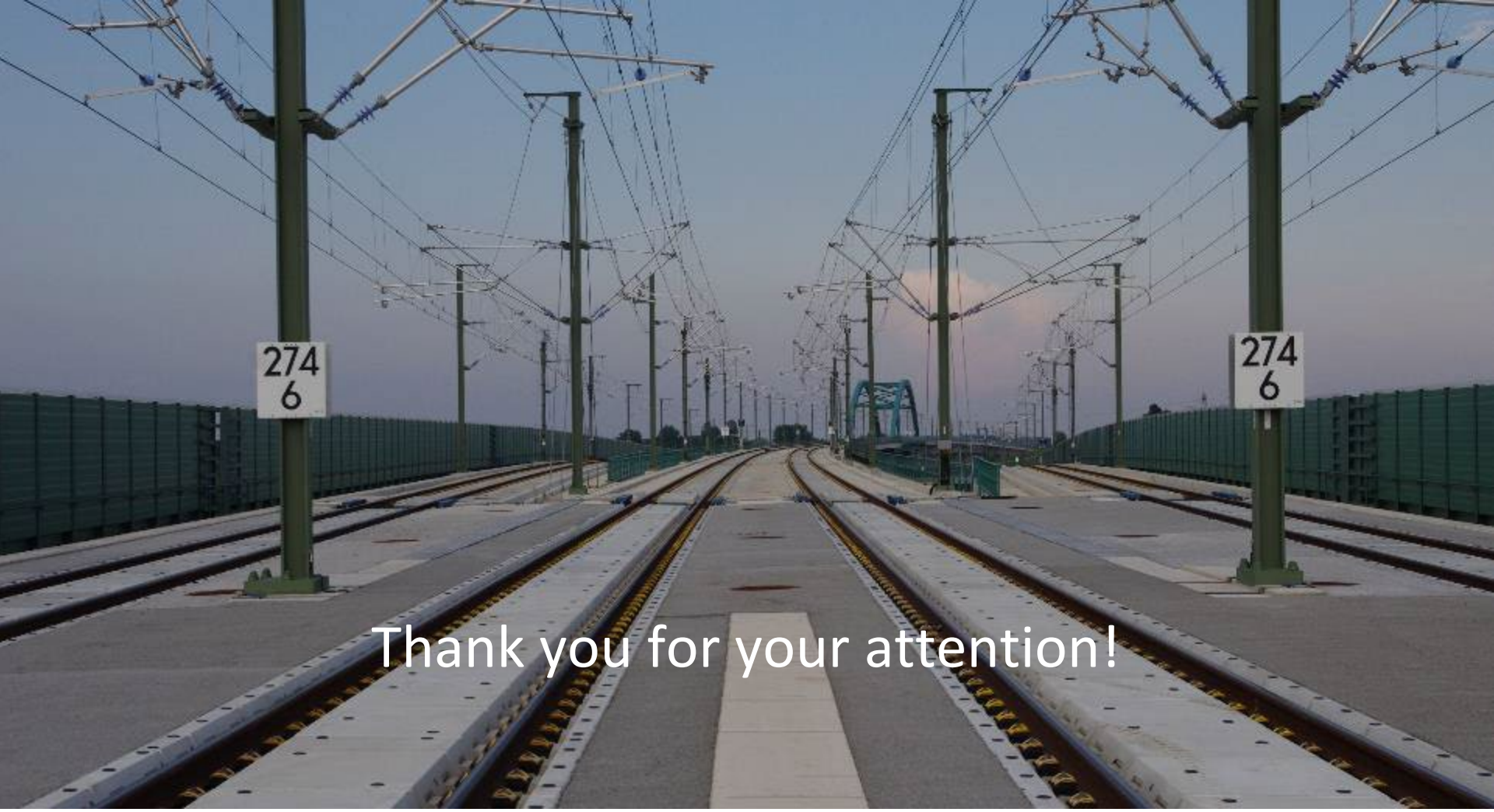
In railML 3.3 we will add state for elements in other railML subschemas as well.

Three possible solutions:

1. Add <rollingstockStates>, <interlockingStates> etc.
2. Move to <common>/<states> and keep reference from <state> to elements
3. As 2 but reverse reference, so elements point to their <state>

[Issue #491: Element state in Common]





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