Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Fri, 28 Dec 2018 20:39:08 GMT
View Forum Message <> Reply to Message

Dear all,

I'm joining the discussion a bit late, as I have spent the past six months recovering from a head injury and I still only work part-time.

It seems I agree with the majority (everyone?) on option 1 on both questions. Regarding references, there are some redundant cross-referencing on the same level, e.g. <NetElement>s and <NetRelation>s both reference each other. But that is maybe inherited from RTM?

I agree with Martin Karlsson that there are some peculiarities with the structure and naming in the IL subschema. Regarding the prefix verbs he mentions, I cannot see that these are useful as Jörg v. Lingen claims. The usefulness of such verbs is that they describe the relation between the element and its parent. The schema is full of good examples that such verbs are useful. But in the cases of <ownsTvdSections>, <knowsRoutes> etc. who owns and knows? I cannot see that any of the "owns" and "knows" under EntityIL contribute to the interpretation of the element names, and hence they should be skipped.

It is clear that "the views in IL are different". That does not necessarily mean that the views in IL are wrong, but then the design rules should be adapted. In <common> there are containers outside of a view (electrificationSystems, speedProfiles), only one view (positioning) and one hybrid (organizationalUnits). N <infrastructure> there are also three containers outside of a view (physicalFacilities, infrastructureVisualizations and infrastructureStates). Applying the current design rules to IL, I would say that all of the IL "views" behave as containers outside of a view (with a possible exception for assetsForILs which behaves as a container of views). I will make some further comments in a separate thread in the IL subforum.

The xsd design pattern was not mentioned in the presentation, but I will include a comment here anyway. Mostly, it currently looks like a Garden of Eden pattern, but the local and global element names do not match. Mostly, it is just a matter of capitalization, but there are also a lot of cases where the difference is more substantial (mostly in IL, but also in IS). And, as XML is case-sensitive even differences in capitalization matter. Also, there is no point in generating global elements for types that are never used in any local elements such as abstract types. The pattern should be to define elements also globally, not to generate elements for all types. Currently there are some global elements that cannot be used in valid xml (because they implement abstract types), some that should not be used (because their names are not similar to any local elements), and no local elements are really defined globally (case-sensitivity + other differences).

Despite these critical comments, I think we have come a long way, and the RC looks very promising. We are just not at the finish line yet.

Best regards,
Thomas Nygreen
Jernbanedirektoratet