Subject: Formation versus TrainParts
Posted by Stefan de Konink on Wed, 12 Feb 2020 16:08:41 GMT
View Forum Message <> Reply to Message

We are currently implementing an open source converter from NeTEx (the European CEN standard for the exchange of network and timetables) and RailML 2.4. While having produced XSD-valid railML 2.4 files, the (new) software supplier of our launching customer complains about the way we have modeled portion working.

The ground truth is being exported from Giro Hastus is CEN NeTEx. We are in control of the (open source) OIG script used for this step. In order to model portion working we are using NeTEx VehicleServices to differentate between tasks a vehicle has to operate. A task begins and ends at a depot, making it analogue to a NeTEx Block. For each task we know the NeTEx VehicleType. To improve our data quality for the traveler perspective we integrate VehicleTypes into the more specific NeTEx CompoundTrain which is virtually analogue to a RailML formation.
Our source data does not split vehicles within a single commercial trip.

In our implementation we differentiate between formations for sections that trains run coupled. An example would be a long train splitting into two short trains. Each formation has been assigned an individual RailML Block matching a NeTEx VehicleService. Where the RailML Block references to RailML blockPart which are shared between blocks. This allows us to create 1:1 relationship between a blockPart, trainPart and train having an unambiguous RailML train, including a formation. Important detail: we are not duplicating trainParts or trains, each train has a single trainPart, always having a single formation.

The new software vendor insists on a different model: every subsection of the long train should be a unique and duplicated (so not just referenced to a trainPartRef as in TT_Rostering.xml example) down to the lowest common
formation. Trains would be consisting of trainParts with a short_a and short_b formation. The fact that short_a and short_b is formation_long, is lost.

Main modelling differences:

our model has three blocks, referencing to 1 single blockpart, referencing 1 single trainPart. "This is per block what should be executed"
their model creates two rosterings (long, short). Long is operated shared, short individually. Having in total three blocks, referencing to unique blockParts, referencing to unique trainParts (duplicating the timetable). "This is per train/trip what it consists of"

Given that both of our models (except for their private attribute) match the XSD validator, and our model can loaded into railVIVID without errors. Can we expect that our producer interface could be certificated as-is? If our method is not compliant with the scheme RailML proposes, could anyone state why grouping by formation is a bad practice?