Subject: [railML 3.2] How to handle different granularity in node-edge-models at the level="micro"?
Posted by Karl-Friedemann Jerosch on Wed, 08 Dec 2021 15:50:51 GMT
View Forum Message <> Reply to Message

Dear railML users,

as a node-edge-model can be described in railML by using the topology elements "netElements", "netRelations" and "networks",
it is possible on the one hand (option 1) to define in railML a topology whereas the nodes are only switches, buffer stops and other kind of project topology borders.
On the other hand (option 2) it is also possible to define in railML a topology whereas axle counters are considered as nodes additionally to switches, buffer stops and other kind of project topology borders.

How should data exchange between two (or more) tools be managed, if one (or more) is working with option 1 and the other one(s) is running with option 2?

Subject: Re: [railML 3.2] How to handle different granularity in node-edge-models at the level="micro"?
Posted by Milan Wölke on Mon, 04 Apr 2022 14:19:53 GMT
View Forum Message <> Reply to Message

Hi Karl-Friedemann,

from my point of view it should work in a way, that one first matches the funtional infrastructure elements, the ones that have an identity that exists across systems. It would be best to do this at least with the ones that have a spotLocation. Like this it should be possible to distinguish between elements that exist in the railML and in the importing system, the ones that do not exist in the railML anymore and the ones that are new in the railML. Next, one would search for the nearest elements known in both models of each element known in both models (I would define nearest based on the importing systems data). With this the mapping of the two graphs should be possible and based on this all other functional elements should be possible to be placed either on existing edges or on edges that are new in the incoming railML.

Another more obvious approach would be to follow the track-graph that exists on top of the netElement, netRelations graph. Of course this would only work if the necessary trackBegin and trackEnd information exists in the incoming data.

My 3 cents. Hope it helps some. I would really be interested in your experience with this algorithm? What was your approach so far?

Best regards, Milan