Subject: Feedback from 1st railML 3.1 Workshop 09./10.01.2018 - spot locations
Posted by christian.rahmig on Wed, 07 Feb 2018 21:19:13 GMT
View Forum Message <> Reply to Message

Dear RTM colleagues,

on January 9-10, 2018 the first railML 3.1 Workshop took place in Berlin. The aim of this workshop was to collect feedback on the beta version of railML 3.1 that has been released in Octobre 2017. As railML 3.1 is based on RailTopoModel V1.1 (November 2017) another question that has been raised deals with the multiplicity of coordinates in SpotLocation to be forwarded to you:

A SpotLocation defines the location of a NetEntity on the basis of a single point within the topology network. A NetEntity (e.g. an OperationalPoint) may have an arbitrary number of SpotLocations in order to reference it with different coordinate systems or to define different "application points" of the NetEntity in the topology network.

Example in railML 3.1 beta:

```
<operationalPoint id="opp01" ...>
  <spotLocation id="opp01_sloc01" netElementRef="ne01" intrinsicCoord="0">
    <geometricCoordinate positioningSystemRef="gps01" x="14.3269"
y="49.0896"/>
  </spotLocation>
  <spotLocation id="opp01_sloc02" netElementRef="ne01" intrinsicCoord="0">
    <linearCoordinate positioningSystemRef="lps01" measure="231.860"/>
  </spotLocation>
</operationalPoint>
```

The problem:
Currently, a SpotLocation allows only for referencing exactly one location coordinate, e.g. a WGS84 coordinate or a (national) mileage coordinate or screen coordinates (for CAD drawings). This means, that if a location shall be described in different coordinate systems, several SpotLocation elements have to be defined. This has been considered by the workshop participants being not very beneficial, because of redundancy (repeating attributes @netElementRef and @intrinsicCoord). Instead, it was suggested to allow for referencing more than one location coordinate within the same SpotLocation element as long as the same location is adressed (unchanged parameters @netElementRef and @intrinsicCoord).

The following modified example shows how the problem can be solved if the RTM structure will be adapted:

```
<operationalPoint id="opp01" ...>
```

```
  <spotLocation id="opp01_sloc01" netElementRef="ne01" intrinsicCoord="0">
    <geometricCoordinate positioningSystemRef="gps01" x="14.3269"
y="49.0896"/>
    <linearCoordinate positioningSystemRef="lps01" measure="231.860"/>
  </spotLocation>
</operationalPoint>
```

Questions resulting from the discussion:
* Is there a reason why currently a SpotLocation allows only for referencing one location coordinate instance?
* What do you think about the proposal above of having more location coordinate instances for one SpotLocation as long as they refer to the same physical location? If considered positively, when will such a modelling change be implemented in RTM (V1.2?)?

Thank you very much and best regards
Christian Rahmig

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Subject: Re: Feedback from 1st railML 3.1 Workshop 09./10.01.2018 - spot locations

Posted by Airy Magnien on Thu, 12 Jul 2018 15:03:17 GMT
View Forum Message <> Reply to Message

Two questions, two answers:

A1: Indeed, under RTM 1.1, a SpotLocation is related to exactly one positioning system coordinate instance. Not sure there is a "reason" for that, as it is definitely possible to use, concurrently, several positioning systems:
Any given Located Net Entity can reference an unlimited number of entity locations. Let us assume some Entity would be indeed be located, physically, at a certain spot. It is possible to instantiate several SpotLocationCoordinate objects characterizing this physical spot, each one related to a coordinate itself related to another positioning system. In other words, if n positioning systems are of interest and if there are m "spotlike" entities to locate, you could instantiate (m*n) SpotLocationCoordinate objects to fulfil your needs.
The obvious drawback is, it seems uneconomical to instatiate n*m spot location objects when m such objects could do the job. Not sure this would be a real problem, though.
Semantically, the "solution" above is not very clean: if the net entity is associated with several spot locations, nothing tells whether these locations are intended to be coincident. Odds are, when using coordinates from different positioning systems, that they will never coincide exactly (mathematically). This could create an ambiguity, especially if precision and tolerances are not managed.

A possible improvement would consist in sub-classing (or not, to be discussed) SpotLocationCoordinate for holding 1..* references to positioning system coordinates. The implicit rule would be that each coordinate must be associated with a different positioning system, lest a redundance (or, worse, an inconsistency) would appear; for the time being we do not include OCL rules in our modeling, so the constraint should simply be documented in a note, and its enforcement would be user (or developer) responsibility.

A2: Answer 1 above suggests that we are favourably inclined.