Subject: Visualization: Proposal to move to a separate subschema
Posted by Thomas Langkamm on Wed, 02 Sep 2020 13:30:11 GMT
View Forum Message <> Reply to Message

One topic recently discussed in the infrastructure subgroup was whether the visualization subscheme should be moved from infrastructure (IS) to a separate subschema. Another question was whether the schema needs to be extended. An example where this might be helpful is a TvdSection (part of the interlocking schema).

Here is my proposal how to handle this, which is of course completely up for discussion.

I propose to move the railML visualization to a new subschema, that can freely reference all other schemas.
Reasons: From a technical standpoint, anything in the infrastructure scheme shouldn't reference something outside of the infrastructure schema. The infrastructure schema is pretty much the basis for all other schemas, and from a database/object-oriented standpoint (normalization), there should be no circular references. For example, as interlocking has references to infrastructure, there should be no references from infrastructure to interlocking. If we want to allow visualization to reference all other subschemas, it needs a separate subschema.

At this time, I don't think we need to extend/change the structure of the visualization schema.
Reasons: The visualization schema is already very generic. For those who are not familiar with it, it allows the representation of arbitrary elements either as a spot, a line or an area. Any railML object can be referenced by an visualization entry that includes a type (spot, line/polygon or area/closed polygon), coordinates (1 tuple (x,y,z) for a spot, at least 2 tuples for a line/polygon and at least 3 tuples for an area/closed polygon). Additionaly, each visualization object can include a reference to a symbol.

This approach is completely generic and not restricted to infrastructure. But it may only provide layout information to programs that know "how to draw" objects. In the example of TVD sections, it would be feasible to describe a TVD section as a polygon (closed or not), maybe encircling/following the track parts covered. It could alternatively be a spot object, say a circle with the name of the TVD section and possibly other information, where the visualization schema only contains the center point of the circle.

As a consequence, there is a lot of variability how the visualization schema is used. For infrastructure, it seems natural to describe track (net elements) as lines/polygons, so that a track plan could be drawn completely from the information contained. But for more advanced objects, it's up to the user how much information is part of the visualization schema and how much additional information is "known" by the drawing program. (Case in point: We wouldn't want to describe exactly how a timetable program draws its graphical timetables. So I don't really see that we need the visualization schema for timetables.)

I would, however, suggest to slightly extend the visualization schema as follows:

If we allow spots and polygons for representations, it seems natural to allow circles and ellipses too. Thus, we might add ellipticProjection to the existing types linearProjection, spotProjection and areaProjection. Attributes/structure would be identical to spotProjection, except that it has 1..2

coordinates and a diameter (a circle is defined by a center point and a diameter, an ellipse by 2 focal points and a diameter).

I would suggest to allow an explicit grouping of visualization objects. For example, a large track plan may be split into a large number of visualizations, each depicting a part of the network. Naturally, some parts of the network would appear in several visualizations. The general idea is that we would optionally define "pictures" (not sure if this is the ideal name), and each visualization object can have a reference to 0..1 picture(s).
I have not yet worked out a proposal how to formally add this to the schema, but could do this if the community agrees that this extension makes sense.