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.

Subject: Re: Visualization: Proposal to move to a separate subschema
Posted by on Mon, 07 Sep 2020 08:38:35 GMT
View Forum Message <> Reply to Message

Dear Thomas,

in general, I welcome and agree to your approach. I have no objections against
- move the railML visualization to a new subschema,
- add ellipticProjection
- allow an explicit grouping of visualization objects.

However, I have some minor concerns:

> ...from a database/object-oriented standpoint (normalization), there
> should be no circular references.

This could even be a general thesis. But it is rather a theoretic, "nice to have" rule with not much practical background.
Please be aware that railML is originally designed as a file format for data-exchange, not as a structure for databases. RailML data structures normally will have to be converted into the actual database model (in any) of the certain software at import and converted from such a database model (if any) at export. With that in mind, and as a programmer for import and export of data exchanges with railML, I don't see any practical objections against "circular references" between top-level elements under <railML> as long as there are no actual semantical circular references.

We even already have such circular references and we might have them in future as well. For instance, all aspects of time are a more general phenomenon than infrastructure. Therefore, <times> would have to be excluded from <timetable> and from <infrastructure> into a more general top-level element. But this is not the case in railML and it is probably not practical. Elements such as <timetablePeriod> and <operatingPeriod>s are defined under <timetable> for traditional (and practical, not to write obvious) reasons. They might be referenced from <infrastructure> to express opening/closing times, maintenance periods or the change of

infrastructure during time in general (closure of or creating new tracks, lines, changing permitted speeds, changing track layouts). Of course, <infrastructure> does also need to be referenced from <timetable>. So, we do have "circular references" between <infrastructure> and <timetable> but not in a problematic way.

I still would agree with your thesis but not as a strict rule, rather in a "recommendational", nice-to-have kind.

Best regards,
Dirk.

Subject: Re: Visualization: Proposal to move to a separate subschema
Posted by Thomas Langkamm on Mon, 07 Sep 2020 12:29:03 GMT
View Forum Message <> Reply to Message

However, I have some minor concerns:

> ...from a database/object-oriented standpoint (normalization), there
> should be no circular references.

This could even be a general thesis. But it is rather a theoretic, "nice to have" rule with not much practical background.[...]

I still would agree with your thesis but not as a strict rule, rather in a "recommendational", nice-to-have kind.

I agree. And if I may go off topic just a little bit: In some cases de-normalization (circular references are one example of that, redundant storage of data another) makes a lot of sense. Normalization often increases the complexity of the structures, increasing runtime and making them difficult to understand. And anybody who has been a victim of "overnormalization" -- in a database context it usually means waiting a very long time for some operation to finish, then calculating that it will take X million years to do so, and then ask the developers more or less politely what they have been thinking when designing this :razz: -- will agree that normalization is more a guideline and not an axiom.

Having said this, in this specific I think normalization makes sense and has no drawbacks.

Subject: Re: Visualization: Proposal to move to a separate subschema
Posted by Thomas Nygreen on Fri, 09 Oct 2020 14:09:39 GMT
View Forum Message <> Reply to Message

Dear Thomas,

Apologies for my late reply!

Are there other specific use cases than TVD sections for such a generalisation of infrastructureVisualizations (which is not really a subschema, the railML3 terminology is View)?

The purpose of infrastructureVisualizations is to project the location of physical entities in Infrastructure into a space. Apart from the optional reference to a symbol, it does not describe how to draw the entities, just where they are. In railML3, TVD sections are not considered physical entities (a view I sometimes struggle with, but accept), they are rather considered logical entities that point to physical demarcating entities in the infrastructure. As such, they are easy to place by placing the referenced physical entities. If there are logical entities in Interlocking missing a physical counterpart that can be placed, these should maybe be created in Infrastructure.

If there is need for an elliptic projection, is there perhaps also a need for more general projections? E.g. paths that may have both straight and curved sections (like an SVG path) or volumes in three dimensions?

Best regards,
Thomas
