Subject: Re: [railML3] Handling changes between minor versions
Posted by Thomas Nygreen on Wed, 19 Aug 2020 14:17:05 GMT
View Forum Message <> Reply to Message

Hello and thank you for the feedback!

>> The coordinators are proposing the following three alternatives:
>> 1: Follow the path of railML 2.x in railML 3.x: downward compatibility
>> will be guaranteed. Changes may lead to elements or attributes being
>> deprecated, but not removed.
>> 2: Changes may lead to elements or attributes being deprecated.
>> Elements and attributes that are deprecated in one minor version will
>> be removed in the following minor version. Compatibility is only
>> guaranteed between one minor version and the next.
>> 3: Changes may lead to elements or attributes being removed in a new
>> minor version, without first being deprecated. No> compatibility
>> guaranteed.

> We would prefer option 3 (give up compatibility between minor versions).
> The current compatibility rule often stands in the way of further
> development of the schema and, in our view, has little advantage when
> implementing the interface software.

This is interesting feedback, and it confirms my own suspicions and
experience.

> Furthermore, backward compatibility does not only apply to the removal
> of attributes / elements: Adding a new attribute can also change the
> semantics of other existing attributes, for example, by semantically
> overwriting or negating the contents of other attributes. For this
> reason, it is not always predictable anyway whether a change will break
> downward compatibility or not.

It is possible to achieve backward compatibility in these cases, but
only if you keep using the old attributes in exactly the same way as
before. So even if a new, better modelling is introduced, you will have
to populate all the old elements and attributes as before. It will be
compatible in the sense that when you remove the new attributes and
elements from the file, it will look like it did with a previos version.
But the result of importing that stripped file will most likely generate
a result that does not look like the full model. In many cases there
will be business rules to fill inn the missing information, and these
will probably not match the actual data 100 %.

> However, different revisions of an (already released) version should
> remain backwards compatible.

So far, we have not had updates to already released versions. What kind

of revisions are you thinking about?

Best regards,
Thomas Nygreen - Common schema coordinator, railML.org