Subject: [railML3] Extension methods
Posted by Thomas Nygreen JBD on Fri, 11 Jan 2019 16:01:58 GMT
View Forum Message <> Reply to Message

Dear all,

Apologies for flooding the forum recently, but these are exciting times and there are so many interesting things to discuss.

On page 9 of Christian's slides provided in the modelling patterns thread the "any" extension points are presented. We also know these from railML 2. The effect of these extension points is the flexibility that when a use case arises that is not fully covered by railML, one can create new attributes and elements that can be used at any of these extension points.

In the same thread, Jörg and I briefly touched upon differences between subschemas in the availability of these extension points, but the topic I want to raise here is how these extension points are used.

The complete flexibility of these extension points is both a strength and a weakness. They make it possible to define a new global attribute that can be used on all elements with the anyAttribute extension point. The downside is that it is not possible (with plain XSD 1.0) to restrict where an extension attribute or element can be placed. If needed, this has to be done by formulating rules that are checked through separate routines. One relevant standard for this is Schematron.

The discussion on polymorphism inside railML got me thinking about extensions through polymorphism. In an extension schema it is fully possible to extend a railML complexType to add new attributes. Then, the xsi:type of the standard railML element can be set to this new extended type to include the new attributes. An example:

In railML2.4nor, which uses the extension points provided in railML 2.4, there is a new attribute nor:mounted (I will use the nor: prefix for this extension namespace). This attribute is intended as an extension for the railML element <signal> to describe if the signal is mounted on a pole or gantry. However, there is no technical barrier available to restrict where this attribute is used, so using it on any other element with the anyAttribute extension point will not cause any validation errors. This is not necessarily a problem if we assume that users of this extension apply common sense and read the extension documentation. A use case can also appear where it is useful to use the same attribute on other elements as well. Also, this approach makes it easy to combine multiple extension schemas that were developed independently.

An alternative implementation would be to create a new complexType (e.g. nor:signal) that extends tSignal and adds new attributes such as mounted. To use these new attributes the type of the <signal> element must be recasted by setting it explicitly to the extended type: <signal xsi:type="nor:signal" ... mounted="pole"/>. The strength of this approach is that it does control where the extensions can be used. In this example the mounted attribute can only be used on the signal element and only when xsi:type of that signal is set to nor:signal. The weaknesses are that each element type has to be extended to apply the same attribute to multiple elements, and that it is more complex to combine extensions. If two different extensions are used together, and they both provide a type extending the signal element, one must choose one of the types, or one

schema must base its extension on the other.

It is also possible to combine the two approaches. This reduces the problem of colliding extensions. For technical reasons it is however not possible to add new subelements in an extended type if the original element type contains the any element extension point. For the same technical reasons this problem does not apply to attributes.

The extensions I know (TPS and railML2.4nor) both use the provided extension points and no recasting of elements. For railML2.4nor we have discussed but not yet implemented an extra layer of validation in Schematron.

What are the opinions on extending by recasting (i.e. xsi:type)? Should this be allowed, or even encouraged?
