Subject: Re: XSD design patterns (was: Re: railML 3.x: Data Modelling Patterns)
Posted by Thomas Nygreen JBD on Fri, 11 Jan 2019 17:08:54 GMT
View Forum Message <> Reply to Message

Hi Christian,

Christian Rößiger wrote on Fri, 11 January 2019 17:01
From my point of view, the greater flexibility in reusing the schema and the fact that railML3 apparently already follows the pattern "Garden of Eden" more than "Venetian Blind" (in contrast to railML2.x) speak in favor of a consequnt adaptation of railML3 to the design pattern "Garden of Eden".

I have no experience using Enterprise Architect, but from what I read in the manual, the change away from global elements requires only deselecting one checkbox when exporting. So changing to Venetian Blind is quite effortless. On the other hand, Garden of Eden would require renaming of all existing complexTypes to work according to the pattern's purpose. For many types this is just a matter of different capitalisation, which could maybe be automated, but hundreds of types have other differences as well. Normally, Garden of Eden schemas use @ref instead of @type on elements within types. This ensures equal naming of local and global definitions. When using the option in Enterprise Architect to export global elements for complexTypes to achieve a similar pattern, care must be taken to align type names and element names, and each unique element name must have its own global type of the same name.

Regarding the reuse of elements: I know that this is always listed as a strength of Garden of Eden, but what does it really add compared to extending types from a Venetian Blind schema? Maybe I am missing something. I hope someone can enlighten me. (For Salami Slice this is different, since it does not have global types.)

Quote:
The (subjectively) better comprehensibility of the "Venetian Blind" pattern speaks against this. In addition, as a developer of railML interface software, I do not (yet) see any need for the transfer of subsets of the railML schema, all our current applications are based on the transfer of complete railML files.

The issue that is normally raised against Garden of Eden is that it can be difficult to find the (intended) root element. (But that is a consequence of allowing all elements to be the root.) Apart from that, the two patterns are so similar that I do not find one of them to be more readable or comprehensible than the other. Garden of Eden is just slightly more verbose when viewing the schema as plain text. Although, when listing global definitions in an XML tool, the list will be twice as long for GoE as for VB.

As mentioned, the two patterns are very similar, makimg it quite easy to mix them. Even in a generally Venetian Blind schema, key elements can be defined globally to allow using them as the root. And in a generally Garden of Eden, it is posible to skip global definitions for some elements. For instance, in the railML case, it does not make much sense to use some elements such as a spotLocation as root. Such a mixing of patterns will probably rquire more manual care than

completely adopting one pattern.