Subject: Re: constraints for OperatingPeriod
Posted by on Wed, 17 Oct 2012 18:11:35 GMT
View Forum Message <> Reply to Message

Dear Andreas,

> With the redundancy, I do have a problem. It does allow to be lenient
> when /writing/ railML, but the costs incur at the import side. For
> serious import software, one has to
> - extend the customer-specific specification as to disallow
> inconsistencies between bitmask and rule
> - code a check of the resulting precondition
> - add a test case for the software.

I am aware the problems of redundancies when importing. I do not at all defend them in general.

You do not need to "disallow inconsistencies between bitmask and rule" when importing. You can describe which elements and attributes of RailML are relevant for your software and which are not. (From my opinion, you have to create an own Interface Documentation for a serious import software.) There you can consider either the bit mask or the rule to be relevant and to be exclusive alternatives for your software. If you feel necessary to close all possible inconsistencies you can do, but this is not requested by RailML and probably not completely possible. (For instance, when importing a timetable, you will probably not import all infrastructure and all vehicle data in detail just to check whether there are inconsistencies in it...)

Please consider:
When writing a RailML file, the software does normally not know for which purpose the RailML file will be used. It has to create a RailML file which is most possibly general.
When reading a RailML file, the software can exactly know the requirements of the target system and therefore can decide which elements and attributes are relevant and which additional rules apply. From my opinion, there always will be additional (semantical) rules which are out of the scope of RailML.

RailML does not implicitly avoid all problems that may occur. Rather, RailML gives you a basis not to "reinvent the wheel" again and again each time you need an interface to/from another software.

> So please, let us abstain from casually sprinkle some redundancy through
> the standard for "easer and clearer reading".

Well, ok, but in practice we have to be aware that
- if we always implement the "most scientific, exact but most complicated

way" to describe a case only, to totally avoid redundancies, we risk that RailML will not be accepted by other software developers because of being too much difficult. It should not be our aim to create a RailML which is free from redundancies but nobody will use.
  - sometimes we have to implement a 'short' solution possibly before we ourselves are aware the more general solution. Then, later, when needing the more general solution, we cannot delete the 'short' solution immediately because of the rules of downward compatibility. So, we have a redundancy at least temporarily.

> a concept of unrolling rule-based operating periods onto a calendar,

I agree. This should have been the bit mask.

> and identify "instances" within a period (which is easy once we have
> bitmasks /resulting from unrolling/).

I have some problem with these 'actual' (running) information in a schema called 'timetable'. I think the term 'timetable' implements that it is about planning only, not about 'actual' (running) information.

But, this should not mean that such information should not be possible in RailML. I am not sure whether it is possible to describe the aspects of 'actual' (running) information in a proper way with current RailML. We have some information about that but they are more dead bodies from RailML1...

Anyway, I agree with you: We would need a possibility to identify "instances" within a period to describe 'actual' information additionally to 'timetable' information - either in 'timetable' schema or somewhere else.

>> Additionally, I would prefer to allow an abstract operating period to
>> refer to a 'real' operating period. In my opinion, any abstract
>> operating period earlier or later becomes real.
>
> Here I understand that you aim at the mapping of different stages in the
> planning process.

Yes. I exactly "aimed" on

> a concept of unrolling rule-based operating periods onto a calendar

So far, this should have been the bit mask. But again: RailML is not perfect nor complete...

Best regards,
Dirk.