Subject: Re: Definition of track/stoppingPlace/platform infrastructure vs. timetable
Posted by christian.rahmig on Mon, 07 Oct 2019 18:28:17 GMT
View Forum Message <> Reply to Message

Dear Stefan,

thank you for your contribution on the question "What is a track?".

Am 02.10.2019 um 17:00 schrieb Stefan Hubrig:
> [...]
> <xs:documentation>A Track is defined by a railway section
> between two switches/crossings or between a switch/crossing
> and a buffer stop.</xs:documentation>
>
> Does this definition cover tracks in the context of
> timetables? For those, we would want to describe where the
> train stops inside an operationalPoint. In that case, there
> would be a single line track through many operationalPoints
> since there is no switch. What would <track> <length> refer
> to?
>
> If <track> is not the right fit here, what would we choose
> instead?

Following several talks with different railML users and contributors I can agree that the current definition of a railML <track> seems to be too strict for certain applications/use cases. There are scenarios like yours with very long tracks that span over several operational points and connected switches. On the other side, there are scenarios with very short tracks or "track sections". In both cases, the constraint that tracks range from switch/crossing/buffer stop until switch/crossing/buffer stop, does not match.

My question to the whole community:
Would you like to modify the definition of a railML <track> by removing the constraints?

How to continue then? Is there an optimal definition of a <track>? Here is my proposal:
There may be a hierarchy of <track> elements: A very short <track> section may refer to a longer parent <track> via the attribute @belongsToParent. And this longer <track> may refer to a very long parent <track> spanning over several operational points again via the attribute @belongsToParent.

The advantage of this approach: we are able to model all kind of tracks - from very short to very long. The disadvantage: more freedom on the model requires more constraints on the use case side in order to guarantee compatibility of export and import interfaces. In particular, each use case shall define what kind of <track> elements it expects.

Dear community, how do you like this proposal?

> For most timetable applications it is sufficient to know on
> which "track" of the operational point the train will stop
> (or pass). But a more specific description could be either
> of:
> •   stoppingPosition (currently not in railML) Describes where the
> front of the train stops
> important parameters for compatibility with a train: train
> type/category, direction
>
> •   stoppingPlace
> Refers to the train stop position with the length
> important parameters for compatibility with a train: train
> type/category, direction, train length
>
> •   platform
> Important for passenger trains.
>
> So what do we choose when the meaning of <track> in
> infrastructure is something different? More generally, when
> do we use track, platform or stoppingPlace?

Sorry, but I did not get the difference between a "stoppingPosition" and a "stoppingPlace". railML 3.1 already knows the <stoppingPlace> element, which defines the place where trains may stop (with their head). Further, you may define the stop post panel by using the <signal> element with child element <isStopPost>.

The <platform> element is relevant for use cases related to the exchange of passengers in a station. For example, the height of a platform edge may be relevant for evaluating the access of the train with wheel chairs. Another example may be the side of the platform edge in relation to the orientation of the track in order to tell the passengers the right side of exit. To cut the story short: the choice between elements <platform>, <track> and <stoppingPlace> depends on the use case specific requirements. The railML data model provides the syntax for all different solutions.

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)