Subject: railML 3.x: Data Modelling Patterns
Posted by christian.rahmig on Fri, 18 May 2018 07:01:45 GMT
View Forum Message <> Reply to Message

Dear all,

upcoming new major version railML 3.x will contain lot of modifications
and changes. In order to keep a unified layout and model structure, some
elementary "Data Modelling Patterns" have been defined. These patterns
shall represent a set of rules for modellers (and users) how certain
things are being structured in the model. You can find a first version
of these "Data Modelling Patterns" in [1].

I would like to get your feedback on this approach: Do you agree with
the overall concept of having Data Modelling Patterns? What do you think
about the different patterns in particular?

Any reaction is highly appreciated...

[1] file replaced by newer version on August 17th, 2018

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by christian.rahmig on Fri, 17 Aug 2018 14:51:19 GMT
View Forum Message <> Reply to Message

Dear all,

there is a new draft version of the "Data Modelling Patterns" document
available in [2]. The following small changes have been incorporated:
* adding pattern "Names of elements and attributes"
* clarification on Element vs attribute
* clarification on preferred solution for pattern "Boolean information"

Any reaction is highly appreciated...

[2] file replaced by newer version on January 16th, 2019

Best regards

Christian

PS: To avoid confusion, the old file [1] has been deleted.

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Am 18.05.2018 um 09:01 schrieb Christian Rahmig:
> Dear all,
>
> upcoming new major version railML 3.x will contain lot of modifications
> and changes. In order to keep a unified layout and model structure, some
> elementary "Data Modelling Patterns" have been defined. These patterns
> shall represent a set of rules for modellers (and users) how certain
> things are being structured in the model. You can find a first version
> of these "Data Modelling Patterns" in [1].
>
> I would like to get your feedback on this approach: Do you agree with
> the overall concept of having Data Modelling Patterns? What do you think
> about the different patterns in particular?
>
> Any reaction is highly appreciated...
>
> [1]
> file replaced by newer version on August 17th, 2018
>
>
> Best regards
> Christian
>

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by on Wed, 14 Nov 2018 15:05:26 GMT
View Forum Message <> Reply to Message

Dear all,

In anticipation of the next TT developer meeting I have dealt with the above document and would like to document the points relevant to me in the following.
Christian, please don't see it as a negative comment ;-), with all not mentioned points I agree and I generally welcome that there are defined modeling principles.

I see a general problem: The TT data cannot be easily classified into microscopic or macroscopic. This is probably due to the fact that the TT objects are not hierarchically composed, but rather networked on the same level (m:n relations). Example: I have the option to specify a departure time to the minute or to the second or to omit it at unimportant stations, but from my point of view it is rather a question of vagueness than of macroscopic / microscopic. In all cases, the same data type is used (xs:time), which is always processed the same way by the reading system.
The only obvious differentiation according to microscopic/macroscopic results from the dependencies to the railML3 infrastructure.

Comments on the individual chapters:

Model hierarchy and inheritance (slides 3-6)

Do the requirements for container elements also apply if the container is a subelement (part) of another object or only at the top level?
Otherwise I see the requirements already fulfilled in railML2.x-TT:
- There is a general base element tElementWithNameAndId for all TT objects
- Top-level containers contain only elements of type

Element vs. Attributes (Slide 14)

The mere presence of several (countable) attributes is not a criterion for choosing element vs. attribute. With this reason one could also model a coordinate (x, y) as a countable attribute with the help of two elements. More important seems to be the existence of further information about the respective attribute (validForDirecton="both" in the example).

Unknown information and Default-Values (slide 15/16)

I understand the intended rule to mean that the absence of an attribute should always be interpreted as an "unknown value" in the future. I think this rule is too strict and has several disadvantages.
Currently, a missing attribute can have the following meanings:

1) Value unknown
2) Value intentionally absent. Example: arrival / departure (time) for beginning / ending trains at first / last station
3) Does not apply in context. Example: <stopDescription>.onOff only makes sense when held commercially, otherwise the meaning is "does not apply". However, this value does not currently exist for "onOff

4) Static default value is to be used. Example:
<ocpTT>.trainReverse="false" applies in 99% of cases and can therefore be regarded as the default value.
5) Context-dependent default value must be used. Example:
<stopDescription>.stopOnRequest is implicitly always true for non-commercial holds and therefore does not have to be explicitly specified.

Possible consequences:

1) is still allowed
2) and 3) could only be eliminated by changing the modelling. I see there primarily the replacement of attributes by elements, since a missing element can still represent both "value unknown" and "value intentionally not present". It is questionable whether this makes the schema more understandable.
4) and 5) These actual default values can certainly always be specified explicitly in the future, but in my opinion this would lead to an insane enlargement of the files. At least the (more frequent) static default values (4.) should be preserved.

Boolean information (slide 17)

It seems to me that the case "Mandatory Boolean with true|false as valid values" is not intended. What is the reason for this limitation?
In general, I would look at boolean values as general attributes and not define them as a special case. Option 2 therefore contradicts the statement "Mandatory elements can never be unknown" (slide 15).
In addition, the question arises in which cases a boolean value will be used in the future and when an enumeration with 2 elements will be used.
In my opinion, this decision has so far been decided more according to semantic aspects.

References (slide 21)

As I said before, I still don't have the idea to divide the TT-world into macroscopic / microspcopic, but so far option 1 is used (macroscopic refers to microscopic) more frequently, which I still think makes sense. Example: A <train> references its operating points (<ocpTT>) and not vice versa.

Multiplicity (slide 24)

I generally agree with the statements, whereby in special cases there may be container elements with a minimum of 2 elements, e.g. there is currently a <Path> element with minOccurence="2" in railML2.x.

What is missing from my point of view

Definition of container elements in xml/Uml.

Currently most containers are defined in railML using xs:sequences. Is there a "ban" on other structuring elements such as "xs:choice" or "xs:all" or their equivalents in UML by railLM3-IS?

Use of polymorphism

Inheritance of elements is currently already used in the schema, but rather by defining "template elements" with certain general attributes. However, there are hardly any cases in TT where the schema requires an (abstract) base class, and alternatively different classes derived from it can be used. This would be conceivable, for example, when mapping hold information with an abstract element <ocpTT> with some common attributes and derived subclasses <ocpTTPass>, <ocpTTCommercialStop> or <ocpTTOperationalStop> with additional individual attributes. Is there any rejection or encouragement on the part of railML3-IS?

Best regards
Christian Rößiger

--
iRFP e. K. · Institut für Regional- und Fernverkehrsplanung
Hochschulstr. 45, 01069 Dresden
Tel. +49 351 4706819 · Fax. +49 351 4768190 · www.irfp.de
Registergericht: Amtsgericht Dresden, HRA 9347

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Joerg von Lingen on Mon, 19 Nov 2018 04:25:05 GMT
View Forum Message <> Reply to Message

Dear all,

Christian's proposal for the modelling pattern is quite useful to normalise the sub-schemas within railML. Basically I
agree with the set patterns. However, it shall be used as a guideline and not as a strict corset. The railway world is
wide spread and requires flexibility.

I see currently the following issues where the strict adherence to the patterns is not useful:
1) Hierarchy
RS: The view would be formation/vehicle which are at the same time containers. Seeing

Engine/Wagon/Maintenance/Classification/... as objects their parts definitely have several levels. One could argue to
use the components as containers within the "super"-container vehicle. But this would bring a lot overhead in referencing.
IL: According the pattern we have again "super"-containers without real view-level - AssetsForIL/Controller/SignalBox/GenericIM. The objects in these containers sometimes need more than one part-level.

2) Layer
IL: Elements like Controller or SignalBox cannot live without the remaining AssetsForIL/GenericIM because they need them
as basis.

3) Extension points
RS: In the 2.4 version there are not much extension points requested. At least most of enumerations can be extended.
IL: A good portion of elements are derived from EntityIL which provides the possibility of any-elements. However,
extensions with any-elements were not requested. The majority of enumerations cannot extended as this is mostly not
sensible.

4) Booleans
RS/IL: I would prefer option 1 and make the attributes in question optional.

5) Naming
IL: The use of verbs in the names enhance the legibility as it points already to the related function of the element. In
addition it would cause conflicts or confusion, if a name like "overlap" appears at several locations due to the
referencing.

Thanks to Christian for his work.

Jörg v. Lingen - Rollingstock/Interlocking coordinator

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Cédric Lavanchy on Mon, 19 Nov 2018 11:49:08 GMT
View Forum Message <> Reply to Message

Dear all,

I find a good idea to have some rules how elements have to be modeled. These are rules, but rules are made to be broken. Not in all cases but I can imagine, that at some place, they could be counterproductive. My advise is to tolerate some difference to the rules but it must be justified and known from the community, otherwise it is the open door to any kind of abuse.

Here some specific points:

1) "Inheritance": I have nothing against inheritance except that it is not always correctly use. See the Liskov substitution principle (Square should not inherit from Rectangle)

2) "Common domain": take care to put in there only the required elements and not everything, otherwise it will become impossible to manage

3) "Names of elements and attributes": I heard many times some complains about the size of the files and for instance by the Timetable, effort are made to reduce the amount of date in order to reduce the size of the file. On the other hands, the names can become longer. We have to take care not to ruin some efforts.

4) "Boolean information": option 1 (optional version)

5) "References": option 1 (if they are not sub-elements already)

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by on Thu, 22 Nov 2018 10:28:33 GMT
View Forum Message <> Reply to Message

Dear Christian and all,

there is already a lot of reply for this topic, so I reduce my words (this time ;-)) to a short summary concerning two issues which are very important for me:

1) Hierarchy

I think a rather flat hierarchy has no advantage. I prefer a good (possibly deep) hierarchy especially in a very 'technical' context. I already have often the problem of needing to 'jump' very often in the railML files (when reading manually) to resolve references.

Additionally, when I made the suggestion of a possible generic model for future <TT> (with a very flat hierarchy), it was widely refused because of too less structure. So, I am probably (obviously) not the only one with this opinion.

2) Default values

I want to encourage what Christian Rößiger wrote. We cannot avoid default values. It is highly unreasonable for everyone to write "operationalStopOrdered=..." each time when there is no operational stop up to the horizon. Same applies e. g. to "guaranteedPass=..." which does not make sense when there is no pass but a stop. I would simply be more confusion than helping for someone who reads the railML file.

We have plenty of such examples in <TT>.

Best regards,

Dirk.

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Joachim.Rubröder on Tue, 27 Nov 2018 15:08:12 GMT
View Forum Message <> Reply to Message

Dear all,
I generally agree with the design patterns, as long as we really try to respect them within all railML 3.x domains in order to create a common look and feel for railML.

1) "Hierarchy:" Ok, Container/Object/Parts (or Components) would correspond with trains/train/trainParts. Views could be "commercial trains", "operational trains", "ordered slots", ...

2) "Naming": Ok, so there will be lots of boolean attributes like "isThis" or "hasThat"

3) "Boolean information": Option 1, a dedicated "unknown" should be part of an enumeration together with "assumed", "irrelevant", ...

4) "References": Option 1, because the linking might not be as straight forward as intended. A train could belong to several different kind of train categories.

Best regards,
Joachim

--
SMA und Partner AG
Gubelstrasse 28, CH-8050 Zürich
www.sma-partner.com

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Martin Karlsson on Tue, 11 Dec 2018 12:48:52 GMT
View Forum Message <> Reply to Message

Looking at the Interlocking domain in the v 3.1 release candidate, I have observed a number of deviations from these rules. As the rules are quite new, it is still an open discussion whether to change the rules or the schema, or even keep the rules but motivate a deviation. Either way, I would like to highlight my findings.

1) Hierarchy. IL has introduced a "super-container" level (e.g. AssetsForILs) between the Domain (Interlocking) and View (AssetsForIL). So you can have multiple Views of the same type in one file.

The reason for this is apparently that Controller, SignalBox and SpecificIM are considered to be Views (and they obviously exist in multiplicity). But in my opinion, they are Objects. A view should represent a certain type of information, not an individual entity.

I believe this should be aligned so that it is the same everywhere. Either follow the rules, or change them.

2) Naming of containers. According to the rules, container elements are named from the objects they contain, but in plural. In addition to this, IL has added a preamble, qualifying the type of relation. E.g. "knowsRoutes" instead of just "routes".

I'm not convinced that this preamble adds any value to the developers. As far as I have seen, there are not multiple containers of the same object type anyway. But also here, an alternative is to apply this pattern consistently.

3) Naming of Objects. This is not really covered by the rules, but anyway... The principle used in IL has been to first try to find a name that is different from what is used in IS (e.g. MoveableObject instead of Switch). If this has not been possible, a suffix of "IL" has been added to the name (e.g. LevelCrossingIL).

I would prefer to use the suffix variant consistently. That would make it apparent that the IS and IL objects are different views of the same thing. In earlier drafts of the IS domain, this principle was used to distinguish between functional and physical assets.

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Joerg von Lingen on Wed, 12 Dec 2018 05:02:13 GMT
View Forum Message <> Reply to Message

concerning 1) the views in IL are different. You shall think about the different levels in interlocking:
control level with HMI and traffic management systems - Controller
interlocking level with safety related logic - SignalBox
field element level with physical assets - AssetsForIL
SpecificIM is an addition to view the specific rules

concerning 2) I still see the preambles useful. There might be other ways but would require extensive rework of the schema. This may be done anyway concerning polymorphism as stated below.

concerning 3) we have in-between decided to have always the suffix in case of name collisions in different domains. The suffix shall be used in both domains if developed during one release. Otherwise only the later coming gets the suffix. Thus we will have SwitchIS and SwitchIL etc.

With MoveableObject we have another issue. It is not SwitchIL but the abstract class SwitchIL is based on. At the moment the inheritence is used by xsi:type, however, I understood now this may not be the best way in XML world.

Best regards,
Joerg v. Lingen - Interlocking Coordinator

On 11.12.2018 13:48, Martin Karlsson wrote:
> Looking at the Interlocking domain in the v 3.1 release
> candidate, I have observed a number of deviations from these
> rules. As the rules are quite new, it is still an open
> discussion whether to change the rules or the schema, or
> even keep the rules but motivate a deviation. Either way, I
> would like to highlight my findings.
>
> 1) Hierarchy. IL has introduced a "super-container" level
> (e.g. AssetsForILs) between the Domain (Interlocking) and
> View (AssetsForIL). So you can have multiple Views of the
> same type in one file.
>
> The reason for this is apparently that Controller, SignalBox
> and SpecificIM are considered to be Views (and they
> obviously exist in multiplicity). But in my opinion, they
> are Objects. A view should represent a certain type of
> information, not an individual entity.
>
> I believe this should be aligned so that it is the same
> everywhere. Either follow the rules, or change them.
>
> 2) Naming of containers. According to the rules, container
> elements are named from the objects they contain, but in
> plural. In addition to this, IL has added a preamble,
> qualifying the type of relation. E.g. "knowsRoutes" instead
> of just "routes".
>
> I'm not convinced that this preamble adds any value to the
> developers. As far as I have seen, there are not multiple
> containers of the same object type anyway. But also here, an
> alternative is to apply this pattern consistently.
>
> 3) Naming of Objects. This is not really covered by the
> rules, but anyway... The principle used in IL has been to
> first try to find a name that is different from what is used
> in IS (e.g. MoveableObject instead of Switch). If this has
> not been possible, a suffix of "IL" has been added to the
> name (e.g. LevelCrossingIL).
>
> I would prefer to use the suffix variant consistently. That
> would make it apparent that the IS and IL objects are
> different views of the same thing. In earlier drafts of the
> IS domain, this principle was used to distinguish between
> functional and physical assets.

>
>

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by christian.rahmig on Tue, 18 Dec 2018 21:17:48 GMT
View Forum Message <> Reply to Message

Dear all,

thank you very much for your feedback on the Modelling Design Patterns. As we are breaking new ground with these modelling design rules, we are thankful for every single feedback, no matter whether pro or con.

As an intermediate result of the discussion, we can conclude that the defined Modelling Design Patterns will be handled as guidelines and not as written law. This means for the modelling that we try to follow these patterns whenever it makes sense, but for sure there will be cases, when we will deviate from these patterns for certain reasons. These reasons must not be single schema internal ones, but shall be analyzed and discussed cross-schema wide. By doing so, we want to ensure that deviations from the Modelling Design Patterns are not schema specific, but have reasons that are considered by other developers, too.

As someone said at the last Conference in Prague: "You are not going to design modelling rules, but harvest them." Following this approach we want the modelling design patterns to learn and to evolve over time in parallel to our data model. Consequently, every feedback is highly appreciated and will make the railML 3.x better step by step.

Thank you very much and best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Fri, 28 Dec 2018 20:39:08 GMT
View Forum Message <> Reply to Message

Dear all,

I'm joining the discussion a bit late, as I have spent the past six months recovering from a head

injury and I still only work part-time.

It seems I agree with the majority (everyone?) on option 1 on both questions. Regarding references, there are some redundant cross-referencing on the same level, e.g. <NetElement>s and <NetRelation>s both reference each other. But that is maybe inherited from RTM?

I agree with Martin Karlsson that there are some peculiarities with the structure and naming in the IL subschema. Regarding the prefix verbs he mentions, I cannot see that these are useful as Jörg v. Lingen claims. The usefulness of such verbs is that they describe the relation between the element and its parent. The schema is full of good examples that such verbs are useful. But in the cases of <ownsTvdSections>, <knowsRoutes> etc. who owns and knows? I cannot see that any of the "owns" and "knows" under EntityIL contribute to the interpretation of the element names, and hence they should be skipped.

It is clear that "the views in IL are different". That does not necessarily mean that the views in IL are wrong, but then the design rules should be adapted. In <common> there are containers outside of a view (electrificationSystems, speedProfiles), only one view (positioning) and one hybrid (organizationalUnits). N <infrastructure> there are also three containers outside of a view (physicalFacilities, infrastructureVisualizations and infrastructureStates). Applying the current design rules to IL, I would say that all of the IL "views" behave as containers outside of a view (with a possible exception for assetsForILs which behaves as a container of views). I will make some further comments in a separate thread in the IL subforum.

The xsd design pattern was not mentioned in the presentation, but I will include a comment here anyway. Mostly, it currently looks like a Garden of Eden pattern, but the local and global element names do not match. Mostly, it is just a matter of capitalization, but there are also a lot of cases where the difference is more substantial (mostly in IL, but also in IS). And, as XML is case-sensitive even differences in capitalization matter. Also, there is no point in generating global elements for types that are never used in any local elements such as abstract types. The pattern should be to define elements also globally, not to generate elements for all types. Currently there are some global elements that cannot be used in valid xml (because they implement abstract types), some that should not be used (because their names are not similar to any local elements), and no local elements are really defined globally (case-sensitivity + other differences).

Despite these critical comments, I think we have come a long way, and the RC looks very promising. We are just not at the finish line yet.

Best regards,
Thomas Nygreen
Jernbanedirektoratet

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Fri, 28 Dec 2018 21:14:06 GMT
View Forum Message <> Reply to Message

Joerg von Lingen wrote on Mon, 19 November 2018 05:25
3) Extension points

RS: In the 2.4 version there are not much extension points requested. At least most of enumerations can be extended.
IL: A good portion of elements are derived from EntityIL which provides the possibility of any-elements. However,
extensions with any-elements were not requested. The majority of enumerations cannot extended as this is mostly not
sensible.

In 3.1-RC EntityIL does not offer an any-elements extension point. It just offers any-attribute. Consequently, no type in IL offers the possibility for extension elements.

Best regards,
Thomas

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Joerg von Lingen on Sat, 29 Dec 2018 04:08:51 GMT
View Forum Message <> Reply to Message

Thomas,

is this a request to have any-elements extensions? If so where are they needed?

Best regards,
Joerg v. Lingen

Rollingstock Coordinator

On 28.12.2018 22:14, Thomas Nygreen wrote:
> In 3.1-RC EntityIL does not offer an any-elements extension
> point. It just offers any-attribute. Consequently, no type
> in IL offers the possibility for extension elements.
>
> Best regards,
> Thomas

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Thu, 03 Jan 2019 17:50:21 GMT
View Forum Message <> Reply to Message

Joerg von Lingen wrote on Sat, 29 December 2018 05:08
Thomas,

is this a request to have any-elements extensions? If so where are they needed?

If I knew now, I would argue for including the elements in the IL subschema, rather than having to use extensions. It is in the nature of future use cases and user needs that we do not foresee them. Of course, there can also be elements that should not be included in railML, but are useful or necessary in a context-specific extension; I do not know if such examples exist already. Requests will probably come when the new version is put into use. In railML 2.x there was initially few extension points, but in the end they were introduced almost everywhere.

Currently, the IS subschema has any-elements built into all entities, while IL does not allow it at all, but it does allow any-attributes. I find that odd and inconsistent. I suggest to apply the same strategy as in IS in all subschemas.

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Thu, 03 Jan 2019 20:22:40 GMT
View Forum Message <> Reply to Message

Dear all,

One more comment regarding naming of subelements and attributes. In general, I find that most attributes and subelements are named in a self-explanatory way, adhering to the described patterns. But there are a few exceptions, both regarding not repeating element names in attribute names and using verbs in boolean attributes. The repetition rule in Christian's presentation does not apply to elements, but in some cases it should. For instance, the IL element <ownsATPdevice> has a child element <atpDevice> (also a case of inconsistent capitalisation of abbreviations).

Regarding naming of boolean attributes, most without a verb are in IS:

common3.xsd
(Line 685) @negated

infrastructure3.xsd
(Line 1422) @virtual
(Line 1432) @switchable
(Line 1471) @cleaning
(Line 1476) @fueling
(Line 1481) @loadingFacilities
(Line 1486) @maintenance
(Line 1491) @parking
(Line 1496) @preheating
(Line 1501) @ramp
(Line 1506) @toiletDischarge
(Line 1511) @waterRestocking
(Line 1516) @sandRestocking
(Line 1521) @electricSupply
(Line 3113) @regenerativeBrakingAllowed (switch suffix verb to prefix)

(Line 3122) @complianceTSIRequired (switch suffix verb to prefix)
(Line 3192) @powerLimitationRequired (switch suffix verb to prefix)
(Line 3207) @automaticDroppingDeviceRequired (switch suffix verb to prefix)
(Line 3252) @switchOffBreaker
(Line 3257) @lowerPantograph
(Line 3271) @switchOffBreaker
(Line 3276) @lowerPantograph
(Line 3281) @changeSupplySystem

interlocking3.xsd
(Line 178) @autoNormalisation
(Line 622) @automaticKeyRelease
(Line 627) @automaticKeyLock
(Line 1594) <automatic> (NB: boolean element)

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Fri, 11 Jan 2019 16:09:32 GMT
View Forum Message <> Reply to Message

I have been thinking more about the extension points, but in a wider perspective than discussed here, ao I decided to move it to a new thread.

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by christian.rahmig on Wed, 16 Jan 2019 09:26:39 GMT
View Forum Message <> Reply to Message

Dear all,

thank you for all your comments and discussions on the railML 3 Data Modeling Patterns. I considered your remarks and also added some new issues into a new version of the document. It is available in [3].

Any further comments and reactions on it are still highly appreciated and will make railML 3 better for application.

[3] http://forum.railml.org/userfiles/2019-01-16_railml_railml3- modelling-patterns.pdf

Best regards
Christian

--
Christian Rahmig - Infrastructure scheme coordinator
railML.org (Registry of Associations: VR 5750)
Phone Coordinator: +49 173 2714509; railML.org: +49 351 47582911
Altplauen 19h; 01187 Dresden; Germany www.railml.org

PS: To avoid confusion, the old files [1] and [2] were deleted.

Subject: Re: railML 3.x: Data Modelling Patterns
Posted by Thomas Nygreen JBD on Mon, 21 Jan 2019 13:56:15 GMT
View Forum Message <> Reply to Message

In slides 37 and 38 I think the top-level/global element <tracks> should also be removed for "Venetian Blind". With this style, the only global element will be <railML>.
