# Problem statement

## Origin / sources

railML.org (C. Rahmig, Infrastructure Coordinator) has requested several changes to be considered via the railML.org forum (RTM section). The suggested changes are summed up into one alternative diagram (see annex; the diagram is also on sharefile: 2018-07-02\_railml\_railml3-induced-changes-to-rtm12.pdf).

The rationale is contained in various parts of the forum and copied or summarized in the text below.

The sections are:

## Context

railML3.1 beta 2 release is expected for September.

## Problems

The suggested changes address six issues; these are treated separately in the present document. Compiler (A. Magnien) has added his personal opinions and comments, basing on other concurrent works. The comments have partly been published in the forum, while drafting the present document.

## Discussion

There were two levels of discussion:

* Sharefile workflow launched by UIC (expires 29.8, midnight)
* Webconf with C. Rahmig, railML.org coordinator for infrastructure, 29.8

## Decisions

The proposed decisions reflect the above discussions.

## Deferred

Designates interesting ideas resulting from the discussions, but not foreseen to be included in RTM 1.2 for at least one of the following reasons:

* + No apparent use case
  + There is a use case, but solution is not needed to issue railML3.1
  + There is a use case, but proposal is not mature enough

*Note: herebelow, the ‘@’ character prefixes attribute identifiers, but is not expected to be used in the RTM class diagram.*

# Problem 1: Adding offset attributes

## Problem statement

From the thread “Lateral location of elements”:

Locating elements next to the railway track can be done on the basis of a LinearCoordinate element and its parameter @lateralOffset. By defining that negative values shall be handled as lateral distances to the left (in the direction of NetElement orientation) and positive values vice versa to the right, the exact location of the element is specified.  
  
Question: How can I model an element situated left or right of the track without knowing the exact distance? For example: For opening the doors of the train on the correct side, I just need to know at which side the platform edge is situated. From my current understanding of the RTM it is not possible to model this reduced information without using pseudo values.

## Proposed Solution 1

From the thread “Lateral location of elements”:

The information about the side and the lateral distance should be separated in two parameters (e.g. @lateralSide, @lateralOffset) so that it is possible to define the lateral side without a specific distance.

## Comments

There is a need to describe the infrastructure irrespectively of its geometry (that’s why we have topology in the first place), so it’s basically OK. *Added: a typical use case is schematic signaling plan.*

Proposed solution works, but needs further clarifications and choices:

Solution 1.1 🡺 @lateralSide is mandatory. @lateralOffset is expected to be positive in the direction given by @lateralSide. An OCL rule may be introduced to force a value >= 0.

Solution 1.2 🡺 @lateralOffset sign gives the side: (-) always means left, (+) always means right. This allows to have @lateralSide optional. If the attributes contradict each other, @lateralOffset is ignored (variant: an OCL rule raises an exception).

## Decision

* **Solution 1.1 shall be implemented, i.e. side and positive “offset” shall be provided** (“at minus one meter on the left” does not sound right, does it?)
* **Semantic clarification: “offset” may be positive or negative, but not used in the model; instead:**
* **Attributes @lateralSide and @lateralDistance are introduced, replacing @lateralOffset, where @lateralDistance is essentially positive. Resulting rules are: both attributes are optional; if @lateralDistance is provided, it shall be positive or zero, and @lateralSide must also be provided**
* **Attributes @verticalSide (values: up, down) and @verticalDistance also to be adopted, as suggested in the workflow, for reasons of consistency.**

## Deferred

* The homogeneous and complete (from an engineer’s point of view) description of units and values, including distance for the present case. Consistency with IFC to be sought.

# Problem 2: Redundant reference to Positioning System

## Problem statement

From the “Reference to PositioningSystem” thread:

In RTM every PositioningNetElement contains at least one AssociatedPositioningSystem. The AssociatedPositioningSystem references a PositioningSystem. In railML 3.1 the matching XML syntax looks like this:  
<netElement>.<associatedPositioningSystem>@positioningSystemRef  
Further, an AssociatedPositioningSystem contains at least one IntrinsicCoordinate, which is typically linked with a PositioningSystemCoordinate. Each PositioningSystemCoordinate itself references a PositioningSystem. In railML 3.1 the matching XML syntax looks like this:  
<associatedPositioningSystem>.<intrinsicCoordinate>.<\*coordinate >@positioningSystemRef  
This way of modelling seems to result in some redundancy w.r.t. referenced positioning system.

## Proposed Solution 2.1

From the “Reference to PositioningSystem” thread:

Therefore, my question I would like to answer together with you: Is it really necessary having the first reference (<netElement>.<associatedPositioningSystem>@positioningSystemRef) being mandatory? I think that the second reference (<associatedPositioningSystem>.<intrinsicCoordinate>.<\*coordinate >@positioningSystemRef) is sufficient. Therefore, I suggest to either remove the first reference or make it optional in the model. What do you think?

## Comment

While there is an apparent redundancy, the class “AssociatedPositioningSystem” becomes meaningless if the associated positioning system is not designated.

*Not totally in fact, since the properties “valid from… valid to” characterizes a set of intrinsic coordinates, and this validity period*

Besides, redundancies have pros (e.g. consistency checks) and cons (e.g. storage space). railML did not state what actual problems this redundancy brings.

## Decision

**Role ‘positioningSystem’ of class AssociatedPositioningSystem shall be made optional (cardinality 0..1 instead of 1). This leaves the possibility to omit redundant info. However:**

* **We recommend to use this redundancy for consistency checking;**
* **We demand not to mix coordinates referring to different positioning systems in one same composition, because this is why the class “AssociatedPositioningSystem” has been introduced in the first place.**

# Problem 3: Validity times

## Problem statement

From the “Validity times” thread:

the information that a NetElement is valid (for operation) is currently modelled with the attributes @validFrom and @validTo. A resulting small example looks like this:  
<netElement ... validFrom="2018-01-01" validTo="2018-12-31"/>  
This implementation of validity times has two drawbacks:

* It is not possible to model other infrastructure states, e.g. "under construction"
* It does not allow to model segmented validity times, e.g. before and after a construction blocking

The second point is really essential.

## Proposed solution 3

From the same thread:

Therefore, I propose to change the RTM modelling in the following way: instead of attributes @validFrom and @validTo, use a repeatable child element <valid> with attributes @from and @to to define the different segments of validity time. The resulting small example may look like this:  
<netElement ...>  
<valid from="2018-01-01" to="2018-06-29"/>  
<valid from="2018-07-02" to="2018-12-31"/>  
</netElement>

See also the changed class diagram (appendix).

## Comment

Having a complex type for time intervals will become inevitable, in the context of RTM development (BIM, real-time apps…). This is made apparent e.g. by the current IFC time interval modelling.

It will probably become necessary to distinguish the time intervals themselves from their generators (e.g. calendar periodicities) in order to avoid confusion. Train schedulers may want to exchange calendar periodicities, while work programmers will rather rely on the resulting, actual possession times.

Meanwhile, we are not discussing RTM2.0 but RTM1.2. For that purpose, the proposed change may well suffice. It also has the advantage of avoiding multiple definitions of (validity) time attributes in different classes (NetworkResource, AssociatedPositioningSystem, PositioningSystem): where all these attributes are expected to be consistent and comparable, it makes sense to define them once.

## Decision

Somewhat unexpectedly, railML.org stated that the request concerned the IT object lifecycle, rather than the operational availability of an object, as stated in the forum and copied into the text above. railML.org stated that the problem exposed is distinct from the “time dimension” need, with e.g. cyclic availability periods.

IT object lifecycle was not, so far, explicitly modelled (if you refer to IRS30100, you will see that validFrom.. validTo was explicitly foreseen to deal with operational availability).

**Decision deferred on this point – we need a few more days of investigations.**

# Problem 4: Length attribute

## Problem statement

See the thread “Proposal for incorporating length information in RTM NetElement”.

The problem of missing length information in topology network (NetElement) has been discussed once again and a proposal has been derived and communicated in the infrastructure forum [1].The proposal suggests to introduce the parameter @length to the NetElement object. Following this, NetEntity locations have to be adapted (attributes @pos or @fromPos and @toPos), too.

The discussed referenced there illustrates the dilemma of establishing lengths indirectly (over NetEntities such as tracks), which admittedly is cumbersome, or rather directly at topology level, but restricted to the micro level.

## Proposed solution 4

From the same thread:

In this overview, adding attribute @length and renaming attributes for intrinsic position into @pos etc. is marked as issues number 4 and 5.

Precisely, the diagram adds the attribute @length to the class PositioningNetElement.

## Comment

Mixed feelings here:

* Topology does not presume any measure.
* Length makes only sense for linear net elements, not PositioningNetElements in general.
* In the case of switches (at micro level), where do you place the origin?
* In the case of branches (at macro level - figure a double track line), same question.
* At meso or macro level for instance, how would you deal with additivity of lengths, (internal paths inside stations may be several, even between the same entry and exit points). Without additivity, lengths are of little use.

The discussion in the railML.org infra forum reflects most of the interrogations above and restricts the "length" attribute to the micro representation, which makes perfect sense. However, given the recursive structure of RTM, it is difficult to introduce that feature without e.g. adding, via OCL, restrictions on where the attribute should be used.

RTM may be extended at will, as foreseen in IRS30100, so railML.org may well add the length attribute to (linear!) net elements they wish to do so. But conceptually, more work is needed for making the model unambiguous while keeping it consistent.

## Decision

**Proposal is rejected.**

## Deferred

An essential, future subject for cooperation between RTM and railML.org would be the inclusion of track geometry (alignment) in the model.

# Problem 5: renaming intrinsic coordinates

## Problem statement

From the thread “Feedback from 1st railML 3.1 workshop…”:

Intrinsic coordinates are used to define a relative position of a NetEntity between 0 and 1 within the topological network. This attribute is used for all types of locations, e.g. <spotLocation> and <linearLocation>.

The problem:  
The intrinsic coordinate is usually not the "leading" positioning information in data base exports. Instead, it is being calculated on the basis of mileage or meter positioning values and thus represents a "derived" value. The conclusion of the discussion was the statement that for the data exchange intrinsic coordinates are not of interest. Instead, it may be calculated by the importing system based on the positioning information from the input. Thus, it was suggested to make the intrinsic coordinate for all NetEntities optional for the data exchange format since the information seems to be redundant. The concept of intrinsic coordinates in general and within internal data models is not questioned by this discussion.

## Proposed solution 5

The solution seems to be only substantiated in the sketched diagram (see annex).

## Comment

In my understanding, the statement *“Intrinsic coordinates are used to define a relative position of a NetEntity between 0 and 1 within the topological network”* misses the key point (this comment was also made in the forum): in the Topology package, the extremities of (linear) NetElements need to be differentiated (disambiguated) in some way. This is why the corresponding intrinsic coordinates (0 and 1) of PositioningNetElements must refer to some coordinate, in the PositioningSystem package.

Indeed, the relative positions of linear NetElements uses the attributes “positionOnA” and “positionOnB” that are of type “Usage” with values 0 and 1, which are precisely these extremities.

Otherwise, since RTM 1.1, locations do not need to refer to intrinsic positions, since Location::AssociatedNetElement has now two distinct subclasses (-Intrinsic, and -Coordinate), which responds to the need not to compute intrinsic coordinates for intermediate locations.

## Decision

**No change.**

# Problem 6: allow multiple names

## Problem statement

I could not locate the thread where the problem was submitted. In short, the issue is how to represent names in different languages.

## Proposed solution 6

The proposal is very similar to the one made for validity times (see diagram):

* Create a new class Base::Name, with attributes @name and @language
* Base::NamedResource holds 0..\* references to instances of Base::Name
* The corresponding role is “name” 0..\*, not ordered

## Comment

The solution is simple and effective. A few drawbacks:

* It drops the distinction between “name” and “longName” attributes that had been previously introduced in NamedResource.
* As per proposal, it is possible to have several names in one language, but there is nothing to tell which does what.

One may wonder if having qualified roles (to select languages or distinguish long and short names) would not be more realistic, while not really adding complexity.

On the simplification side, one may wonder if “NamedResource” is at all useful, as the role “name” (0..\*) could as well be attached to BaseObject, and the responsibility of naming could be left to the user. What needs to be distinguished anyway inherits from BaseObject and has an ID.

## Decision

**Suggested solution shall be adopted; details to be worked out (qualified roles, language codes…)**

