Changes in RailTopoModel 1.1 compared to RailTopoModel 1.0

Andreas Pinzenöhler, Airy Magnien

November 9, 2017

1. Introduction

RailTopoModel version 1.0 (short: RTM 1.0) has been published by UIC as IRS30100 in April 2016. Since then, RailTopoModel attracted the attention of the industry, of software vendors, and of Universities and Research Centres. In parallel, the early implementers represented in the RTM workgroup continued the testing and implementation of RTM, while also figuring out the future applications and developments. Also, matching successive versions of RTM with successive versions of railML3.x, the associated data exchange format, calls for robust solutions, with some effects on the RTM structure. Overall, these practical works led to a number of small, but significant changes in RTM. These changes are partly documented as notes in the UML diagrams; the following provides more information.

2. Objective of the present document

The present document summarizes and explains the changes relative to the initial version, RailTopoModel 1.0, as published on: `http://www.railtopomodel.org/en/`. The information in dedicated wiki: `http://wiki.railtopomodel.org/index.php?title=Main_Page` will be gradually updated.

3. Overview on RailTopoModel 1.1 main changes and their rationale

The present section mentions most changes following a systematic, top-down approach.

3.1 Design principles: SOLID

According to Wikipedia,

> In computer programming, the term SOLID is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible and maintainable.

RailTopoModel is now published as a standard (IRS30100), for which understandability, flexibility and maintainability are obviously important. The initial version of RailTopoModel has therefore been reviewed in the light of the aforementioned five principles, which we summarize here:

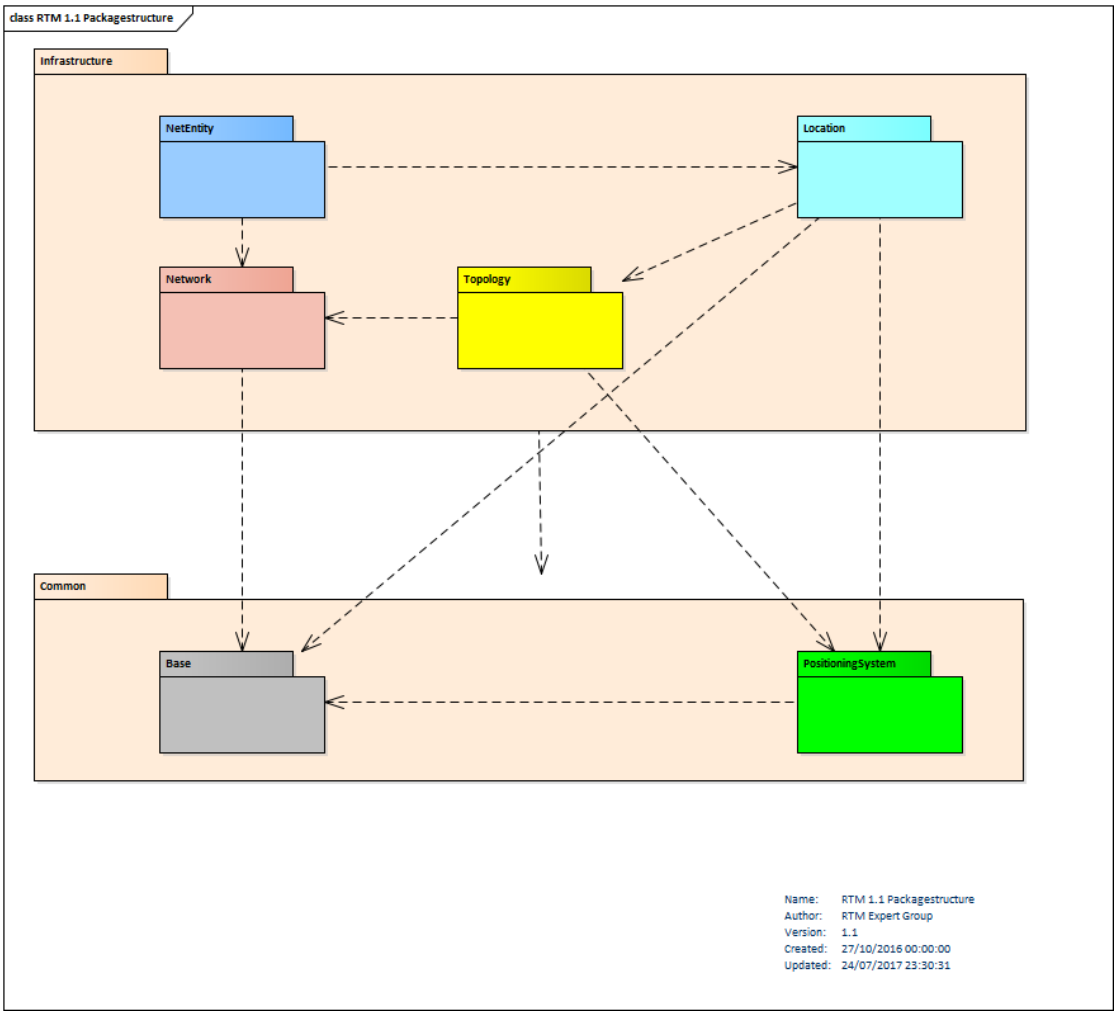
Initial	Concept
S	Single responsibility principle: a class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)
O	Open/closed principle: software entities should be open for extension, but closed for modification.
L	Liskov substitution principle: objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.
I	Interface segregation principle: many client-specific interfaces are better than one general-purpose interface.
D	Dependency inversion principle: one should depend upon abstractions, not concretions.

3.2 Functionalities

RTM 1.1 does not add any new functionality, compared to RTM 1.0. This is only a maintenance release.

3.3 Packages

The overall package structure has been revised. Packages *Base* and *PositioningSystem* have been moved into one "superpackage" called *Common*, and all other packages were grouped into an *Infrastructure*superpackage. The package *Network* has been created to host the classes *Network*, *LevelNetwork* and *NetworkResource*. A few classes, such as *AssociatedNetElement* and *OrderedAssociatedNetElement*, have changed container package.



3.4 Classes

3.4.1 New classes

The former *BaseObject* hosted three main attributes: Identificator, Name, validity period. Most, but not all three are needed in every descendent class. The decision was then as follows:

- *BaseObject* now only retains the identificator, *id*, of generic type *tID*. The identificator type is for the user to choose amongst the daughter datatypes of *tID*: the recommended choice is *UUID*. Users wishing to retain compatibility with existing data repositories may however use *legacyID*.
- The new class *NamedResource* inherits *id* and implements the *name* attribute that was formerly with *BaseObject*. Furthermore, a *longname* attribute has been added for convenience.

- The existing class *NetworkResource* inherits from NamesResource and implements the validity attributes. The semantics of *validFrom* and *validTo* attributes of all descendent classes is now homogeneous: the time period between *validFrom* and *validTo* is the time of functional availability of the network object "in the field".
- Validity periods are also used by other classes in the *Positioning* package. These classes do not derive from *NetworkResource*, but from *BaseObject* or (now) *NamedResource*. The attributes *validFrom* and *validTo* are therefore introduced separately in these classes, with the same name, but different semantics because the validity period does NOT describe the functional availability of an infrastructure element.
- Two new classes, *AssociatedNetElementIntrinsic* and *AssociatedNetElementCoordinate*, have been introduced to separate intrinsic locations from other location concepts. In such way, it is possible to localize use intrinsic locations and (for instance) spatial locations independently from each other.

3.4.2 New attributes in classes

- *BaseObject* now has an attribute *id:ID* in addition to *uuid:UUID*. The optional *id* attribute has been introduced for the purpose of compatibility with existing railML schemes.
- *NamedResource* holds, in addition to the *name* attribute, a new attribute *longname*.
- *LevelNetwork* used to inherit a string attribute *Name* from BaseObject. However, the purpose here was to characterize the level as *Macro*, *Meso*, *Micro* that are defined in the IRS30100. *LevelNetwork* no longer inherits the *Name* attribute, because it is replaced by a specific *descriptionLevel* attribute. We created the corresponding enumeration type to enforce the usage of the standard names *Macro*, etc., for levels. RTM Users are free to extend the enumeration, in line with the possibility offered by IRS30100 to create any additional level when needed.

3.5 Associations and roles

3.5.1 Changed associations

Many small changes took place after the re-definition of *BaseObject* and *NetworkResource*, plus the introduction of *NamedResource* in between.
One major change is the suppression of the direct association between *LinearLocation* and *PositioningNetElement*. Reason is, that association was redundant with the indirect association (via *AssociatedNetElement*) but did not keep the sequence order of the *PositioningNetElement* instances, while this order is absolutely needed.

3.5.2 Naming

Role names are systematically given to the navigable end of any association. Association names (often repeating the target role name) have been removed, because they are not useful (they are not used for generating railML files for instance. Role names are now systematically in the plural if the cardinality is greater than 1.

3.5.3 Cardinalities

Cardinalities have been reviewed and sometimes changed.

3.5.4 Shared aggregations instead of composite aggregations

In RTM 1.0, *Network* instances were composed of instances of *NetworkResource*. In RTM 1.1, these instances can now be shared, as the composite aggregation (black diamond) was changed into a shared aggregation (white diamond). This change enables a flexible definition of overlapping (sub-)networks, a commonly encountered need. Also, the composition link between *Network* instances and network levels such as "macro", "micro", etc. (instances of *LevelNetwork*) has been changed into a shared aggregation, for similar reasons: *LevelNetwork* instances refer to sets of network resources that may, similarly, overlap.

3.5.5 Colour scheme

Each package has one associated colour for representing member classes. The consistency of colours between all diagrams has been checked.

4. Complete list of changes

See appendix (pdf file).