Horizon 2020
European Union Funding
for Research & Innovation

European
Global Navigation
Satellite Systems
Agency

SYSTEM FOR VEHICLE-INFRASTRUCTURE
INTERACTION ASSETS HEALTH STATUS MONITORING

SIA

Project Title:	System for vehicle-infrastructure Interaction Assets health status monitoring
Starting date:	01/03/2018
Duration in months:	36
Call (part) identifier:	H2020-GALILEO-GSA-2017-1
Grant agreement no:	776402

Deliverable D5.3: Modified XSD and matching RailTopoModel class diagram

Due date of deliverable	Month 27
Actual submission date	16.09.2020
Organization name of lead contractor for this deliverable	DLR
Dissemination level	PU
Revision	3.0

Authors

Author(s)	**DLR** Christian Rahmig
	UIC Airy Magnien
Contributor(s)	

HISTORY OF CHANGES		
Version	**Publication date**	**Change**
1.0	07.09.2020	▪ First version
2.0	07.09.2020	▪ Version for review of consortium
3.0	16.09.2020	▪ Incorporating review notes from consortium
4.0	30.09.2020	▪ Incorporating review notes from GSA

Executive Summary

The present document constitutes the Deliverable D5.3 “Modified XSD and matching RailTopoModel class diagram” in the framework of the Project titled “System for vehicle-infrastructure Interaction Assets health status monitoring” (Project Acronym: SIA; Grant Agreement No 776402, www.siaproject.eu).

SIA is a health monitoring system that is installed in regular trains that assesses the condition of assets in both infrastructure and running gear. As such it collects sensor data and derives KPI about the condition of the catenary-pantograph interface and the rail-wheel interface. For connecting the SIA box with external systems, standardized interfaces have to be defined.

This deliverable D5.3 analyses the need for modelling aspects related to the catenary-pantograph and rail-wheel interface on the basis of industry standard approaches. In particular, it aims at **specifying standardized data content for a sufficient digital representation of the railway track and the catenary** / overhead contact line. Standardized data models and data exchange interfaces may directly contribute to a holistic approach of condition-based maintenance system connecting various systems.

The document is structured into three main parts:

- Chapter 1 provides an overview about the **background** for the intended modelling activities. This includes existing data models and standardised exchange formats as well as experiences with their usage and evolution in past and parallel running projects.
- Chapter 2 describes the **data exchange use cases** derived from the SIA system functionality and architecture. The focus of this section is on identifying the required content of data to be transferred via internal and external SIA interfaces.
- Chapter 3 documents the **data modelling approach**. Starting with the agreement on terminology and data modelling patterns, adaptations are done in a platform independent model, from which a modified XSD is automatically generated.

Table of Contents

EXECUTIVE SUMMARY **3**

TABLE OF CONTENTS **4**

ABBREVIATIONS AND ACRONYMS **6**

1 INTRODUCTION / BACKGROUND **8**

1.1 RAILTOPOMODEL / RAIL SYSTEM MODEL 8
1.2 RAILML 9
1.2.1 Introduction *9*
1.2.2 Community based development process of railML® *11*
1.3 PREVIOUS AND PARALLEL ACTIVITIES 13
1.3.1 Project In2Rail *13*
1.3.2 Project IN2SMART *14*
1.3.3 UIC project OntoRail *14*
1.3.4 Shift2Rail project OPTIMA *15*
1.3.5 Shift2Rail projects LinX4Rail and LinX4Rail-2 ..... *16*

2 REQUIREMENTS AND USE CASE **17**

2.1 POSITIONING RELATED MEASUREMENTS 18
2.1.1 Use Case description *18*
2.1.2 Data specification *18*
2.2 CATENARY-PANTOGRAPH RELATED SENSOR OBSERVATIONS 20
2.2.1 Use Case description *20*
2.2.2 Data specification *21*
2.3 WHEEL-RAIL RELATED SENSOR OBSERVATIONS 22
2.3.1 Use Case description *22*
2.3.2 Data specification *23*

3 DATA MODELLING **25**

3.1 PLATFORM INDEPENDENT MODELLING 25
3.1.1 Modelling language *25*
3.1.2 Interfacing with terminology *25*
3.2 TERMINOLOGY 25
3.2.1 Dictionaries and other sources *25*
3.2.2 Ontologies *26*
3.3 MODELLING PATTERNS 26
3.4 DERIVING PLATFORM SPECIFIC MODELS 28
3.4.1 EA internal XSD generation *29*
3.4.2 Post-EA XSD generation from XMI *29*
3.5 PROOF OF CONCEPT 30
3.5.1 The “EA way” *33*
3.5.2 The “XMI way” *37*
3.6 IMPACT ON RTM AND RAILML DEVELOPMENT 40

3.6.1 railML *40*
3.6.2 RSM *41*

4 CONCLUSION AND OUTLOOK **42**

REFERENCES **43**

LIST OF FIGURES

FIGURE 1: RAILML® VERSIONS [39] 10
FIGURE 2: SIA ARCHITECTURE 17
FIGURE 3: COMPONENTS CLASSIFICATION OF CATENARY IN IN2RAIL PROJECT [7] 21
FIGURE 4: COMPONENTS CLASSIFICATION OF TRACK IN IN2RAIL PROJECT [7] 22
FIGURE 5: SIA MODEL 28
FIGURE 6: FOR PIM MODELLING ONLY STANDARD UML CONSTRUCTS (LEFT) ARE BEING USED 31
FIGURE 7: PIM UML MODEL OF TRACK ELECTRIFICATION 32
FIGURE 8: EA PROJECT BROWSER SHOWS THE STRUCTURE OF SIA PIM 33
FIGURE 9: MODEL TRANSFORMATION TOOL IS AVAILABLE IN EA (MENU DESIGN) 33
FIGURE 10: GENERATING PSM USING EA INBUILT MODEL TRANSFORMATION TOOL 34
FIGURE 11: EA PROJECT BROWSER SHOWS THE STRUCTURE OF THE GENERATED SIA PSM 34
FIGURE 12: GENERATED PSM UML MODEL OF TRACK ELECTRIFICATION 35
FIGURE 13: XSD FILE GENERATION TOOL IS AVAILABLE IN EA (MENU CODE) 36
FIGURE 14: EXPORTING XSD USING EA INBUILT XSD EXPORT TOOL 36
FIGURE 15: EXCERPT FROM GENERATED XML SCHEMA FILE 37
FIGURE 16: EXCERPT FROM XML SCHEMA FILE GENERATED BY UML2XSD 38
FIGURE 17: HTML REPORT DOCUMENTATION IS AVAILABLE IN EA (MENU PUBLISH) 39
FIGURE 18: PUBLISHING UML MODEL DOCUMENTATION USING EA INBUILT OF HTML REPORT 39
FIGURE 19: EXCERPT FROM THE HTML DOCUMENTATION REPORT GENERATED BY EA 40

LIST OF TABLES

TABLE 1: PACKAGES OF RAILTOPOMODEL 8
TABLE 2: OVERVIEW OF REQUIRED POSITIONING RELATED DATA 19
TABLE 3: OVERVIEW OF REQUIRED CATENARY-PANTOGRAPH RELATED DATA 21
TABLE 4: OVERVIEW OF REQUIRED WHEEL-RAIL RELATED DATA 23

Abbreviations and acronyms

Abbreviation / Acronyms	Description
ABA	Axle-Box Acceleration
CDM	Canonical Data Model / Common Data Model
CRS	Coordinate Reference System
DLR	Deutsches Zentrum für Luft- und Raumfahrt e.V. (German Aerospace Centre)
DRIMS	Dynamic Railway Information Management System
DUL	Dolce-Ultralite
EA	Enterprise Architect
EPSG	European Petroleum Survey Group Geodesy
EU	European Union
EULYNX	(European initiative/project in the railway signaling domain)
EUPL	European Union Public Licence
IRS	International Railway Standard
GNSS	Global Navigation Satellite System
HTML	Hypertext Markup Language
ID	Identificator
IFC	Industry Foundation Classes
IMU	Inertial Measurement Unit
IRS	International Railway Standard
KPI	Key Performance Indicator
OCL	Object Constraint Language
OCTO	(French IT company)
OGC	Open Geospatial Consortium
OPTIMA	cOmmunication Platform for TraffIc ManAgement demonstrator (H2020 project)
OWL	Web Ontology Language
PHP	PHP: Hypertext Preprocessor (general purpose scripting language)
PIM	Platform Independent Model
POSI	railML use case “Positioning”
PSM	Platform Specific Model
railML	Railway Mark-up Language
RSM	Rail System Model (re-branding of RTM)
RTM	RailTopoModel
S2R	Shift2Rail
SensorML	Sensor Markup Language
SIA	System for vehicle-infrastructure Interaction Assets health status monitoring
SIA_ABA	SIA sub-system “Axle Box Acceleration”
SIA_CDM	SIA sub-system “Component Degradation Model”
SIA_DH	SIA sub-system “Data Hub”
SIA_PANT	SIA sub-system “Pantograph”
SIA_POS	SIA sub-system “Positioning system”
SIA_VP	SIA sub-system “Visualization Platform”

SNCF	Société nationale des chemins de fer français (French National Railway Company)
SOSA	Sensor, Observation, Sample, and Actuator
SSO	Stimulus Sensor Observation
SSN	Semantic Sensor Network
TMS	Traffic Management System
TRL	Technology readiness level
UIC	Union internationale des chemins de fer (International Union of Railways)
UML	Unified Modelling Language
UNIFE	Union des Industries Ferroviaires Européennes ("the Association of the European Rail Industry")
URI	Universal Resource Identifier
W3C	World Wide Web Consortium
XMI	XML Metadata Interchange
XML	Extensible Mark-up Language
XSD	XML Schema Definition

1 Introduction / Background

This chapter introduces into the background of the latter modelling activities. In particular, it provides an overview about the existing data models / exchange format of RailTopoModel (RTM) and railML. Further, previous and currently ongoing project activities related to these models are being summed up.

1.1 RailTopoModel / Rail System Model

RailTopoModel, first released in 2016 by UIC under IRS30100, is a general-purpose railway system model. Its initial scope was the description of the railway network, centred on the topology of the “iron network”, i.e. tracks and lines.

RTM is currently published under version 1.1, a corrective release. It is free for any lawful usage. RTM 1.1 model and documentation can be downloaded from [38].

RTM uses the UML language (class diagrams). It is arranged in packages with strictly separated areas of concerns, as follows:

Table 1: Packages of RailTopoModel

Package	Purpose
BaseObject	Abstract classes, ancestors of most classes; provides basic features such as ID and name
Network	Network (with levels of detail), network resources (abstract)
Topology	Topological relations of network elements (graph model)
PositioningSystem	Geographic and linear positioning (OGC-compliant)
NetEntity	Base classes for functional objects (material, or immaterial such as speed limits) constituting the network
Location	Localizes Net Entities in relation with the network (Topology package) and with positioning systems (PositioningSystem package)

RTM is currently being revised for delivery of version 1.2 (scheduled for 15/12/2020) that contains, inter alia, following additional packages:

- Time axis and project phasing
- Geometry (equivalent to draft IFC Rail track geometry)
- Signalling (including the link with EULYNX Dataprep model)

- Track and Energy subsystems (incl. the link with IFC Rail; not used in the present context)
- Observation & Measurement: a partial transposition, in UML, of the W3C SOSA/SSN ontologies[1], also taking into account SensorML.

RTM 1.2 is not published yet, but its features are made available to the SIA project by courtesy of UIC. RTM 1.2 is also being used by the EULYNX Data Preparation model: [2].

RTM 1.2 is also coupled with other specialized models; such as IFC Rail published by buildingSMART (see [1]).

Model coupling principles were tested and tried in cooperation between RTM and EULYNX, and are shown in the public snapshot [3].

RTM is meant to be used either autonomously for simple usages such as network representation, or in combination with other models providing advanced information about one or more specific railway domains.

Since RTM formalizes fundamental notions (such as topology, positioning, time dimension, etc.) in a robust, abstract, and consistent way, it is a candidate for allowing the representation of the railway system by a “federated” family of models under the ongoing (S2R) LinX4Rail project (see section 1.3.5). In this context, RTM provides both the basic structure and the “glue” ensuring the linking of heterogeneous, pre-existing models. For that reason, RTM was re-branded RSM (Rail System Model), reflecting the fact that network topology is not its only focus. Not to create confusion, we will still use the abbreviation RTM below.

1.2 railML

1.2.1 Introduction

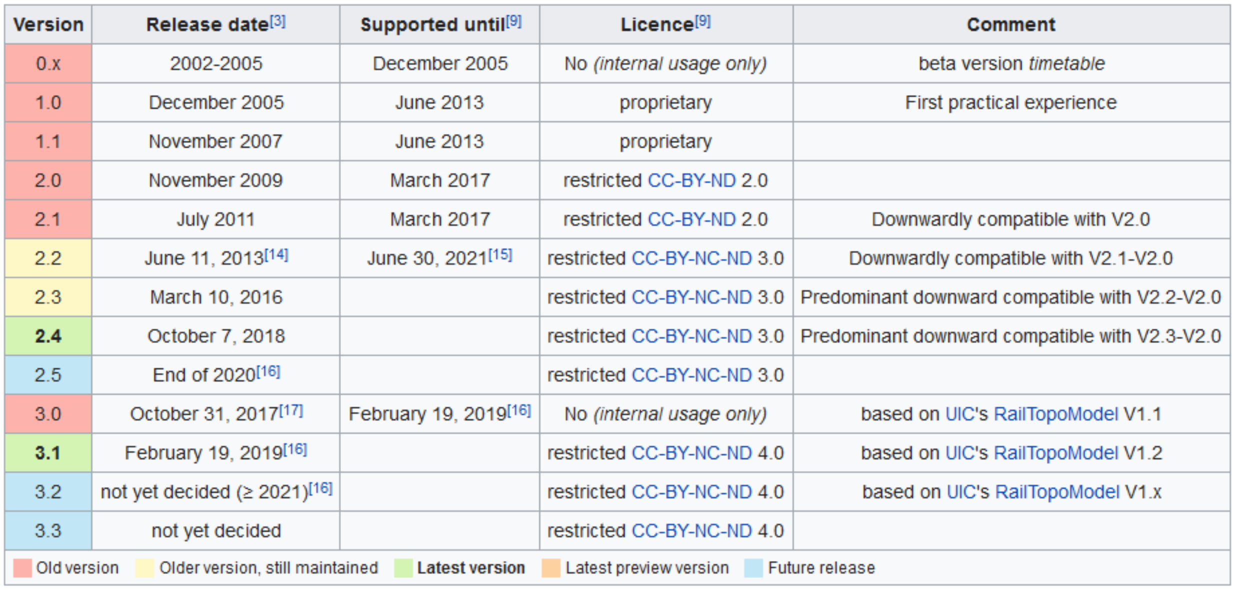
railML® is a data exchange format based on the Extensible Mark-up Language (XML) focusing on railway applications (cp. [11]). At the same time, railML.org is an open source initiative working constantly on the development of this data exchange format for railway applications.

Currently, railML includes the following data schemes (cp. [13]):

- Timetable
- Infrastructure
- Rolling stock
- Interlocking

[1] The W3C SOSA ontology is further discussed in section 3.2.2.

The latest version of railML®, railML 2.4, has been released in October 2018. In parallel, a first version of railML v3, railML 3.1, has been published in February 2019. Both versions, railML 2.4 and railML 3.1, form the current status of the railML standard. The following figure provides an overview about all the versions that are currently supported and the planned future releases (cp. [39]):



Version	Release date[3]	Supported until[9]	Licence[9]	Comment
0.x	2002-2005	December 2005	No *(internal usage only)*	beta version *timetable*
1.0	December 2005	June 2013	proprietary	First practical experience
1.1	November 2007	June 2013	proprietary	
2.0	November 2009	March 2017	restricted CC-BY-ND 2.0	
2.1	July 2011	March 2017	restricted CC-BY-ND 2.0	Downwardly compatible with V2.0
2.2	June 11, 2013[14]	June 30, 2021[15]	restricted CC-BY-NC-ND 3.0	Downwardly compatible with V2.1-V2.0
2.3	March 10, 2016		restricted CC-BY-NC-ND 3.0	Predominant downward compatible with V2.2-V2.0
2.4	October 7, 2018		restricted CC-BY-NC-ND 3.0	Predominant downward compatible with V2.3-V2.0
2.5	End of 2020[16]		restricted CC-BY-NC-ND 3.0	
3.0	October 31, 2017[17]	February 19, 2019[16]	No *(internal usage only)*	based on UIC's RailTopoModel V1.1
3.1	February 19, 2019[16]		restricted CC-BY-NC-ND 4.0	based on UIC's RailTopoModel V1.2
3.2	not yet decided (≥ 2021)[16]		restricted CC-BY-NC-ND 4.0	based on UIC's RailTopoModel V1.x
3.3	not yet decided		restricted CC-BY-NC-ND 4.0	

🟥 Old version 🟨 Older version, still maintained 🟩 **Latest version** 🟧 Latest preview version 🟦 Future release

Figure 1: railML® versions [39]

railML is a user driven standard for data exchange in railways and based on an open development. For interaction between users and developers, a number of tools are provided:

- The **railML Website** is the central point of information. From here, railML users, developers and interested people are directed to all the other information they are searching for (see [12])
- The **railML Forum** is the discussion platform where users can discuss with users and developers about certain modelling and application aspects (see [19]). The different scheme-specific forum topics are moderated by the railML scheme coordinators.
- The **railML Wiki** is the open usage documentation that complements the scheme documentation (see [20]). In particular, railML beginners will find here useful information about the different elements and attributes.
- The **railML Trac** is a ticket system for tracking defects and enhancements that have been discussed and consolidated in the railML forum and which shall be solved / implemented in the future (see [24]).
- **railVIVID** is an open-source tool for validation and viewing of railML files. It can be downloaded from the railML Website (see [14]).

The railML website [15] lists more than 100 companies as railML partners. 24 of these companies are categorized as “developers” including e.g. DLR Institute of Transportation Systems (cp. [16]). 49 companies are categorized as “users”. The remaining companies are tagged as “supporters”.

In order to use railML with projects and products the license terms listed in [17] have to be obeyed. If it is intended to use railML in a productive manner, you are obliged to certify your railML interface(s) that you want to promote or sell. A detailed description of the certification process is given in [18].

1.2.2 Community based development process of railML®

railML® is a community driven project that is coordinated by the non-profit organization railML.org. Consequently, community requests and needs lead the model and schema development.

The following two sections describe the sequence of steps for railML scheme development. The first one is directed to the use case driven railML development, which has been specifically set up for the new railML version 3.x. The second section addresses the process of incorporating small changes in the railML schema as it has been done in the past for previous and current versions of railML 2.x.

railML 3.x: use case view

- **Initial situation**: you have a specific data exchange issue for which you want to use the railML 3.x data exchange format. Such an issue is called a railML use case.
- **Step 1**: Review the lists of use cases in the railML wiki in order to find out whether your use case has already been recorded. A list of requested use cases for railML based data exchange is public available in the railML wiki (see [21]).
- **Step 2a**: If your use case is already listed, review it with respect to your specific task to find out whether there are relevant aspects missing in the use case. If that’s the case, bring your issues to the railML forum (see [19]) and discuss it there together with the railML community. The responsible railML scheme coordinator will lead the process of use case modification and implementation. Finish.
- **Step 2b**: If your use case is not yet listed, bring up the topic in the railML forum to find more partners having the same problem and interest in specific use case implementation. Based on a joint interest, the responsible scheme coordinator will ask you to formulate the use case according to the structured template either directly in the railML wiki or using a Word document. The use case description comprises:
    - A precise description of the data exchange application.
    - A brief analysis of the relevant data flows and interfaces for the data exchange
    - A brief summary of characteristics of the data to be exchanged via the interface.

- **Step 3:** The responsible railML scheme coordinator will initiate a use case working group, which will virtually meet every 3-4 weeks for one hour to work on the use case. The work of the use case working group includes:
  - Consolidation of use case description for updating the linked wiki page
  - Check use case requirements against latest railML version release to identify gaps in the data model and exchange format
  - Discussion of possible solutions for data model extension in order to close the gaps; initiate forum entries to open the discussion also for the general railML community (see [19])
  - Consolidation of the discussion: In particular, required model adaptations are derived and formulated as tickets in the railML Trac ticket system (see [24]).
- **Step 4**: The responsible railML scheme coordinator leads the implementation of the required changes of elements and attributes in the railML data model. Usually, the work is done by the railML scheme coordinator and a specific scheme working group. The implementation comprises:
  - Changes in the railML data model (UML and XSD)
  - Tracking the changes in the railML Trac ticket system
  - Documenting the changes in the source code
  - Documenting the changes in the railML Wiki
- **Step 5**: The use case working group leads the task of use case element specification. The aim of this step is to specify which elements and attributes of the railML data model are mandatory considering the given use case and which elements and attributes are optional.
- **Step 6**: The responsible railML scheme coordinator leads the work of writing an official use case document that brings all the use case facets mentioned before together. The official use case document will be entitled “Use Case Definition” and released on the railML website. Thus, the use case definition is the reference document for certification of railML interface implementations.

<u>railML 2.x: element view</u>

- **Initial situation**: you discover a bug in the existing railML® scheme 2.x or you want to enhance the model at a specific point.
- **Step 1**: Discuss the issue with users and developers of the railML® community in the railML forum (see [19]). There is one forum for each railML subschema and one forum for common aspects.

- **Step 2**: The responsible scheme coordinator summarizes the outcome of the forum discussion and consolidates the solution / result. If the solution comprises a modification or extension of the existing railML data model, the scheme coordinator will create a ticket using the railML Trac ticket system (see [24]). Each ticket is linked with a future version of railML. Thus, users can see when to expect which modifications being implemented in the schema.
- **Step 3**: The railML scheme coordinator leads the implementation of the scheme modification or enhancement and tracks the state in the railML Trac ticket system. The implementation comprises:
    - Changes in the railML data model (XSD)
    - Tracking the changes in the railML Trac ticket system
    - Documenting the changes in the source code
    - Documenting the changes in the railML Wiki
    - Releasing the new version of railML in the railML subversion repository (see [25]) and publishing information about the changes
    - Presenting the summary of changes at the next railML conference
- **In the meantime**: Modifications and enhancements of the railML data model require some time for implementation. If you cannot wait that long, you may want to make use of the “any element” and “any attribute” to attach your own temporary scheme extensions to the railML model.

1.3 Previous and parallel activities

1.3.1 Project In2Rail

The project “Innovative Intelligent Rail” (In2Rail) was meant to set the foundations for a resilient, consistent, cost-efficient and high capacity European railway network. Being a Shift2Rail Lighthouse project running from May 2015 until April 2018, In2Rail explored innovative technologies embedded in a systems framework where infrastructure, information management, maintenance techniques, energy and engineering are integrated and optimised (see [5]).

One main focus of the project was on **Intelligent Mobility Management** including research on automated, interoperable and inter-connected advanced traffic management systems (TMS). In that context a standardised integrated information and communication environment has been set up as a so-called “**Integration Layer**” to provide standard interfaces to external systems outside TMS/dispatching on a plug-and-play basis (see [6]).

The need for data integration and data exchange for TMS purposes resulted in the foundation of a **Canonical Data Model (CDM)**, which was intended as a structured breakdown of the various entities in a railway system focusing on real world objects and relations instead of specific usages or services (see [10]). On basis of the CDM, the Integration Layer should be able to adapt to different TMS external systems and interfaces without running into compatibility problems of data models and exchange formats. The XML based railML is considered as application independent data exchange format that suits to CDM objectives.

Apart from CDM aspects, railML has been identified as appropriate data exchange format covering the needs of modelling static asset status information (see [7]). A data exchange use case related to the needs has been formulated and published for the railML community in its wiki (see [23]). This use case is a direct input to the work in SIA and described in more detail in section 2.

1.3.2 Project IN2SMART

The project "Intelligent Innovative Smart Maintenance of Assets by Integrated Technologies" (IN2SMART) is an EU-funded project within the Shift2Rail framework running from September 2016 until October 2019. It somehow complements the work of the In2Rail project to reach a homogeneous demonstrator for **Intelligent Asset Management** with a Technology Readiness Level (TRL) of 4/5 (see [4]).

One main focus of the project is on designing and implementing a Dynamic Railway Information Management System (DRIMS). In that context, WP 7 of the IN2SMART project aims to develop standard open interfaces to access heterogeneous maintenance-related data. The project deliverable D7.1 (see [8]) provides a review of the state-of-the-art for open data models and formats. As a result of more detailed analysis documented in project report D7.2, the railML data exchange format has been identified as a promising solution for implementation of the standard open interfaces. Further, the need for a CDM based approach that aims at building a canonical railway specific data model independent from applications is underlined (see [9]).

1.3.3 UIC project OntoRail

The UIC Ontorail project was launched in 2019 with the support of its members. The purpose of Ontorail is:

- To progressively document the railway system by describing its "business capability layers", starting from the bottom layers (components, subsystems, networks) and considering the upper layers (operations & maintenance, commercial, business) for future evolutions;
- To derive system descriptions from specialized models that are available in various languages and formats (UML, Express, …) such as IFC Rail (for design & construction) or EULYNX DataPrep, in close cooperation with the responsible organisations (EULYNX, buildingSmart International);
- To publish this descriptive knowledge both as machine-discoverable ontologies and as human-readable, possibly multilingual wikis, resting on proven technology (respectively, OWL2 and Semantic Mediawiki).

Ultimate goals are data exchange (by means of semantic web technologies), which should rest on well-ascertained concepts and terminology to describe railway systems. No standard currently addresses railway systems in a comprehensive way, nor does for instance the UIC dictionary provide definitions, as it mostly provides lexical equivalents across languages. Also, the "standard *replacing* partial standards" is utopic; the proposed, practical solution is to formally collect and link existing partial standards. Specialized ontologies provide the necessary toolbox to precisely relate lexical entries with their underlying concepts, taking into account the context for their usage.

Ontorail has also become one of the components of the S²R Linx4rail project. As a matter of fact, most current railway system descriptions are provided through separate, specialized UML models. Linking such models may happen in two ways:

- At UML level, using inter alia RailSystemModel as a "skeleton" to organize the available models and provide their interconnection;
- At conceptual level, by having each individual model (RSM included) referring to the concepts defined in the Ontorail Ontology.

1.3.4 Shift2Rail project OPTIMA

OPTIMA is a H2020-funded European project coordinated by UNIFE, the full name of which is "Communication platform for traffic management demonstrator". The goals of OPTIMA are[2]:

> *Systems that offer precise, real-time traffic information are fundamental for effective traffic management in the railway sector. Advanced software solutions provide successful traffic management operations. The EU-funded OPTIMA project is designing and developing a communication platform to manage the connection between several services supporting transportation management system (TMS) applications. The platform will link TMS applications with infrastructure systems such as traffic control, maintenance, energy management and signalling. The platform will consolidate real-time data from the railway business service, test the connection of several rail business services with external services and provide a perfectly documented communication platform for supplementary projects. OPTIMA is advanced by a consortium of research institutions, industrial stakeholders and infrastructure managers specialising in traffic control and management.*

Amongst the project objectives, we may highlight[3]:

[2] Source : https://cordis.europa.eu/project/id/881777

[3] Same source

1. use of Integration Layer to integrate real-time data from the rail business service, external sources, services running in the Application Framework and operator workstations
2. definition of detailed data structure according the Conceptual Data Model

Fulfilling the first objective can well be served by a suitable SIA output, especially in the shape of a serialization format. Fulfilling the second objective requires the serialization format to be abstracted into a model or an ontology, in the perspective of integrating it into the Common Data Model that is under construction[4].

1.3.5 Shift2Rail projects LinX4Rail and LinX4Rail-2

Linx4rail, starting December 2019, has a long but self-descripting project title: “System architecture and Conceptual Data Model for railway, common data dictionary and global system modelling specifications”; see [26].

In particular, the ambition of Work Package 3 “Global system modelling specification & development” is to produce a S2R-wide CDM (conceptual data model) and a consistent federation of models, based on existing ones (RTM/RSM, EULYNX, railML and IFC Rail). The modelling efforts in other project such as SIA or OPTIMA are intended to sketch out precursors to the CDM, and to define the principles on which a robust, scalable, and sustainable CDM can be set up.

[4] See : Linx4rail and Linx4rail2.

2 Requirements and Use Case

The requirements and use cases form the basis for the data model content and structure: Use cases – to be understood as data exchange use cases[5] – describe the process of a data transfer from a source via a medium to a destination. Depending on the specific application and environmental constraints, these use cases result in different requirements. This section summarizes relevant use cases and their requirements as direct input for the data model approach in the next chapter.

The following figure depicts the SIA system architecture with its components and interfaces as described in detail in SIA deliverable D2.2 [28].

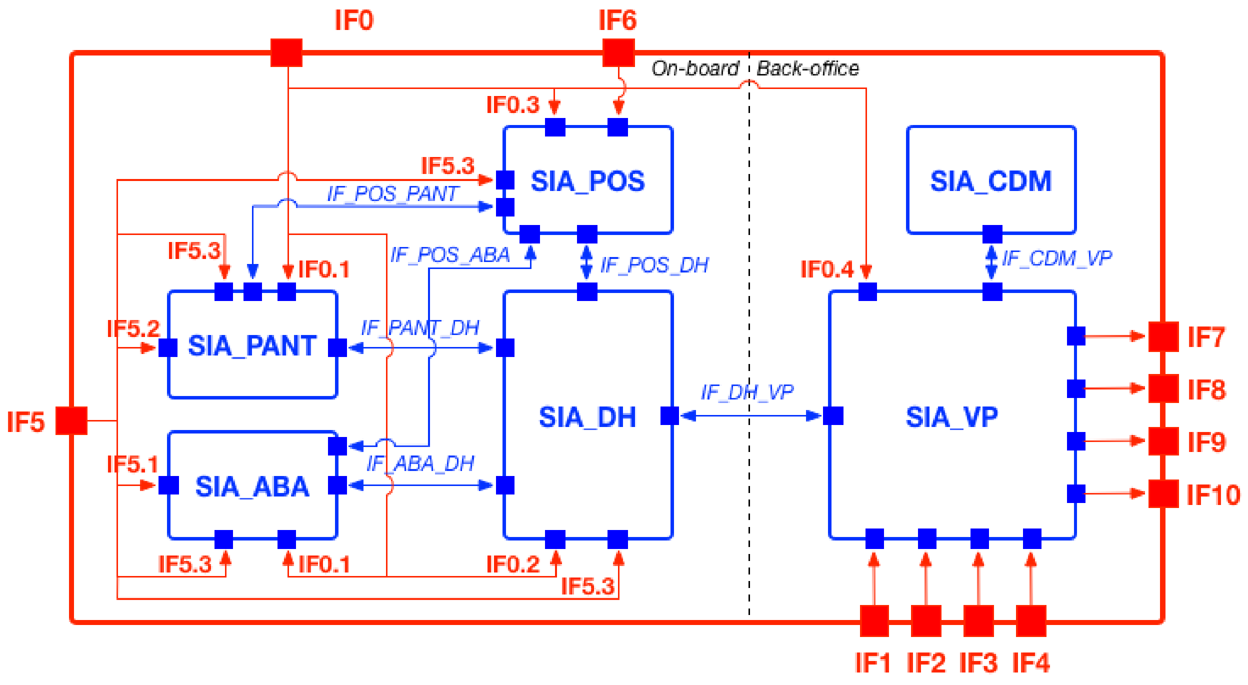


Figure 2: SIA Architecture

The data exchange use cases are realized using the defined interfaces. There are both, internal and external interfaces:

- Internal interfaces between the sensor components (SIA_PANT, SIA_ABA, SIA_POS) and the data hub (SIA_DH)
- Internal interfaces between the SIA_DH and the visualisation platform (SIA_VP)
- External interface IF1 for importing the digital track map with relevant infrastructure components
- External interface IF2 for importing maintenance procedures

[5] The data exchange use case is <u>not equal</u> to the monitoring use cases introduced in SIA D2.1

- External interface IF3 for importing auscultation raw data
- External interface IF4 for importing inspection raw data
- External interface IF7 for publishing asset health status data
- External interface IF8 for informing about early detection of component failure
- External interface IF9 for publishing maintenance recommendations

For data exchange with external systems and services only the external interfaces are of interest. The following sections in this chapter describe data exchange use cases, which may be realized with the related external interfaces, from a data content perspective.

2.1 Positioning related measurements

2.1.1 Use Case description

The focus of this use case is on providing positioning related sensor observations. According to SIA deliverable D3.2 [29] the SIA_POS sub-system components are:

- A multi-frequency GNSS antenna
- A multi-frequency multi-constellation GNSS receiver
- An Inertial Measurement Unit
- A positioning engine at server-side to integrate GNSS and IMU raw measurements with speed and digital map data

For modelling the focus is only on the resulting positioning solution after integration of all sensor observations and maps since this is the potential information of interest for external systems.

The railML.org wiki already includes a data exchange use case “Positioning and Map-Matching” (POSI). Its focus is on train-borne positioning systems based on multi-sensor approaches and a digital track map. The aim is to identify a train or vehicle position within a railway track network defined by the track identifier, a relative position on the identified edge and a direction of travel related to the track orientation (see [22]).

Considering the POSI use case description, the following basic infrastructure focused information shall be modelled for data exchange:

- **Drivable topology**: tracks that are connected with each other in a track network
- **WGS84 coordinates**: to enable referencing with GNSS positions
- **Track coordinates**: to enable referencing with classic railway line positioning system
- **3D alignment** of railway track axis: to enable geometry based map-matching

2.1.2 Data specification

The internal interface **IF_POS_DH** is used for the transfer of positioning related measurements or final positions to the data hub (SIA_DH). This includes:

- IMU data with 10 Hz
- GNSS observation data with 1 Hz
- Positioning output data with 10 Hz
- GNSS timing output with 100 Hz

A detailed overview about the structure and the content of the related interface messages can be found in SIA deliverable D3.2 [29].

For the communication of the positioning output towards external systems the elements and parameters listed in the following table have been selected:

Table 2: Overview of required positioning related data

Component	Parameter	Data type	Description
Track position	Line / track designator	String	Identification of the line / track
Track position	Kilometre position	Decimal (km)	Track kilometre location
Track position	Direction of travel	Enumeration: normal, reverse	Direction of travel related to oriented track / line
Geodetic coordinates	X, y, z	Decimal	Coordinate values in the used coordinate reference system
Geodetic coordinates	CRS	String (URI)	Identification of coordinate reference system, e.g. via EPSG code
3D alignment	Radius	Decimal (m)	Track radius
3D alignment	Gradient	Decimal (per mil)	Track gradient
3D alignment	Cant deficiency / superelevation	Decimal (angle)	Lateral track geometry

2.2 Catenary-pantograph related sensor observations

2.2.1 Use Case description

This use case addresses the communication of catenary-pantograph related sensor observations and KPI. In SIA project deliverable D2.1 [27] three observation and monitoring use cases have been defined:

- Use case #1: Contact wire wear
- Use case #2: Contact wire incorrect height & stagger
- Use case #3: Contact strip wear (normal / asymmetric)

More details about the failure and degradation scenarios forming the basis for these use cases can be found in deliverable D2.1 [27]. The deliverable D4.1 [30] lists the derived requirements for the SIA_PANT subsystem and its interfaces.

From the input documents it can be concluded that the following KPI-related information are relevant for determining the catenary-pantograph related health status:

- **Contact wire geometry**: height, stagger, sag within given limits; decentralisation
- **Contact force** between pantograph and contact wire affecting the dynamic behaviour
- **Pantograph and contact wire wear** calculated from accelerometers (at pantograph) and speed
- **Environmental temperature** affecting the contact wire geometry (sag and height of conductors)

The In2Rail project created a data exchange use case for railway infrastructure asset status representation. This use case (see [7]) focuses on collecting static and dynamic information about the operational status of nine key infrastructure assets including the catenary (also known as overhead line) as central asset for railway electrification. The In2Rail project deliverable D9.1 provides a classification of the asset "Overhead Line" listing all its components. This components classification is depicted in Figure 3 and may be considered as an initial proposal for modelling the catenary as an infrastructure asset.

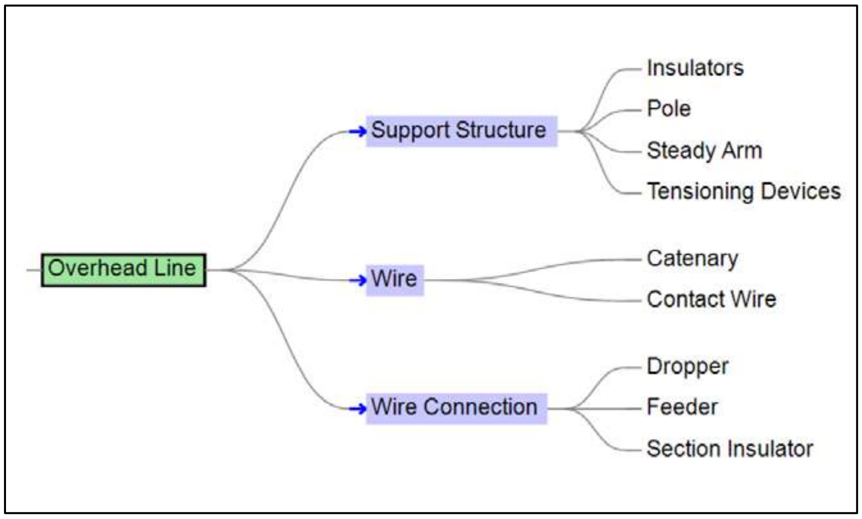


Figure 3: Components classification of catenary in In2Rail project [7]

2.2.2 Data specification

The internal interface **IF_PANT_DH** is used for transmitting condition relevant features and raw data as well as status information on sensor performance from the pantograph sensor module (SIA_PANT) to the data hub (SIA_DH). It is specified in detail in deliverable D4.1 [30].

For the communication of the catenary health status towards external systems and considering the use case description and the input from the In2Rail project deliverable D9.1 the following list of relevant parameters of catenary pantograph related data has been compiled:

Table 3: Overview of required catenary-pantograph related data

Component	Parameter	Data type	Description
Catenary	Track network location	Decimal (km)	Conventional railway linear referencing system
Contact wire	Nominal height	Decimal (m)	
Contact wire	Nominal stagger	Decimal (m)	
Contact wire	Nominal sag	Decimal (m)	
Contact wire	Nominal wire section	Decimal (mm^2)	For dynamic behaviour
Contact wire	Temperature	Decimal (°C)	Environmental temperature
Pantograph	Force	Decimal (N)	For evaluating contact wire

			uplift

2.3 Wheel-rail related sensor observations

2.3.1 Use Case description

The focus of this use case is on the communication of wheel-rail related sensor observations and KPI. In particular, the axle box sensor module (SIA_ABA) collects axle-box acceleration data, which is the source of insights on vehicle track interaction: ABA data can be linked to both, track irregularities (e.g. corrugation, squats, defect joints or hollow sleepers) and wheel shape errors (e.g. flat, polygonal wear and tread spalling). In SIA project deliverable D2.1 [27] three observation and monitoring use cases have been defined:

- Use case #4: Wheel flats and polygonization wear
- Use case #5: Rail corrugations
- Use case #6: Short-wave irregularities

More details about the failure and degradation scenarios forming the basis for this use case can be found in deliverable D2.1 [27]. The deliverable D4.1 [30] lists the requirements for the SIA_ABA subsystem and its SIA-internal interfaces.

On high level, the SIA_ABA sub-system aims at observing these wheel-rail related issues:

- On board monitoring to detect track (wheel) defects.
- Condition of the axle bearings.
- Detect the condition of the wheels profile and the instability of the train.
- Collect data about detached tires.

The already introduced data exchange use case for railway infrastructure asset status representation from the In2Rail project (see [7]) focuses on collecting static and dynamic information about the operational status of nine key infrastructure assets including the track / permanent way. The In2Rail project deliverable D9.1 provides a classification of the asset "Current Track" listing all its components. This components classification is depicted in the following figure and may be considered as an initial proposal for modelling the railway track as an infrastructure asset.

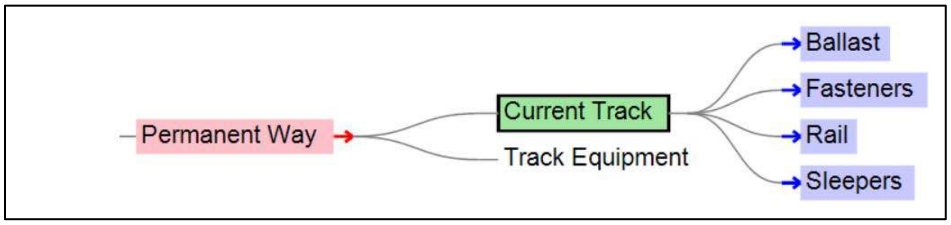


Figure 4: Components classification of track in In2Rail project [7]

2.3.2 Data specification

The internal interface **IF_ABA_DH** is used for transmitting condition relevant features and raw data as well as status information on sensor performance from the axle box acceleration sensor module (SIA_ABA) to the data hub (SIA_DH). It is specified in detail in deliverable D4.1 [30].

For the communication of the track health status towards external systems and considering the use case description and the input from the In2Rail project deliverable D9.1 the following list of relevant parameters of wheel track interface related data has been compiled:

Table 4: Overview of required wheel-rail related data

Component	Parameter	Data type	Description
Track	Track network location	Decimal (km)	Conventional railway linear referencing system
Track	Horizontal alignment type	Enumeration: straight, transition, arc	
Track	Vertical alignment type	Enumeration: hill, valley, straight	
Track	Horizontal geometry: radius	Decimal (m)	For deriving curve resistance
Track	Vertical geometry: slope	Decimal (per mill)	For deriving up-hill resistance
Track	Operational track type	Enumeration: main, secondary, siding/connection	Relates to amount of traffic using this track
Rail	Lateral track position	Enumeration: left, right	To distinguish left and right rail
Rail	Type	Enumeration of rail types	e.g. “UIC 60”
Track	Operational line classification	Enumeration: E5, E4, D4, …	Line classification linked with allowed weight limits
Track	Speed restrictions	Decimal (km/h)	Allowed speeds on the

			track
Sleepers	Type of sleepers	Enumeration: wood, concrete, steel	
Ballast	Type of ballast	Yes / no	No ballast = slab track
Fasteners	Type of fasteners	Enumeration of rail fasteners types	e.g. “K”
Track	Track gauge	Decimal (mm)	e.g. “1435”
Track	(clearance) gauge	Enumeration of gauge profile	

3 Data Modelling

3.1 Platform independent modelling

3.1.1 Modelling language

For the purpose of SIA, a platform-independent model must be set up, open to further re-use, especially in consideration of the LinX4Rail and LinX4Rail2 projects.

The modelling language of choice is, by consensus, UML 2.5 and more selectively, UML class diagrams. Though part of UML, OCL constraints are not widespread and are generally avoided.

3.1.2 Interfacing with terminology

In UML models, the semantics of classes and associations is most often documented in the model itself, using notes that are parts of the UML language, or outside the model, using wikis. The semantics are therefore trapped in text chunks that can be processed by humans only.

Since the first virtue of the SIA model is re-usability, care will be taken to re-use existing models and/or links its terminology (packages, classes, associations, roles) with existing ontologies.

UML is an OMG standard, and rests upon the OMG Meta-Object Facility. This ensures, albeit with some effort, interoperability with other languages that rest on the same meta-model, such as OWL (Web Ontology Language).

3.2 Terminology

3.2.1 Dictionaries and other sources

Accurate definitions of railway terms are provided in various sources. Of relevance to the SIA project are:

- Shift2rail glossary: <in the works>
- IFC Rail: defines classes and property sets of railway subsystems (track, energy, signalling, telecommunication).

Terminology may also be extracted from UML models or XML schema definitions, as required:

- railML2 (xsd) / railML3 (UML)
- EULYNX Data Preparation: the EULYNX Data preparation model has been released online. It is licensed under EUPL 1.1.
- Rail System Model (UML)

None of the above sources of terminology currently have any equivalents in the form of ontologies, although IFC 4.2 (not including the more recent IFC Rail extension) does have such an equivalent, IfcOwl.

The UIC project "Ontorail" and the Shift2rail project "Linx4rail" (Work Package 2, based on Ontorail) envisage the development of a consistent, unique ontology for the railway system based on RTM, IFC, EULYNX, and railML (formal agreement pending on the latter). As of June 2020, the principles underpinning such general ontology have been discussed, but the release of the ontology itself will not take place in time for the present project to re-use.

However, SIA borrowing from the terminology of the aforementioned projects offers sufficient guarantees of future convergence of terminology and will mitigate any impact on the SIA project deliveries.

3.2.2 Ontologies

3.2.2.1 Sensor, Observation, Sample, and Actuator (SOSA)

The SOSA ontology is dedicated to field observations. Quoting from (Krzysztof Janowicz, 2018):

> *The Sensor, Observation, Sample, and Actuator (SOSA) ontology provides a formal but lightweight general-purpose specification for modeling the interaction between the entities involved in the acts of observation, actuation, and sampling. SOSA is the result of rethinking the W3C-XG Semantic Sensor Network (SSN) ontology based on changes in scope and target audience, technical developments, and lessons learned over the past years. SOSA also acts as a replacement of SSN's Stimulus Sensor Observation (SSO) core. It has been developed by the first joint working group of the Open Geospatial Consortium (OGC) and the World Wide Web Consortium (W3C) on Spatial Data on the Web. In this work, we motivate the need for SOSA, provide an overview of the main classes and properties, and briefly discuss its integration with the new release of the SSN ontology as well as various other alignments to specifications such as OGC's Observations and Measurements (O&M), Dolce-Ultralite (DUL), and other prominent ontologies. We will also touch upon common modeling problems and application areas related to publishing and searching observation, sampling, and actuation data on the Web. The SOSA ontology and standard can be accessed at https://www.w3.org/TR/vocab-ssn/ .*

The SOSA/SSN ontology provided the base for the Observation & Measurement package of RSM 1.2.

3.3 Modelling patterns

For the purpose of SIA, a provisional, platform-independent model was established in UML. Pending possible convergence of various models (IFC Rail, railML…), current railML subsystem descriptions were taken as the basis for the PIM.

The model thus includes:

- Common packages imported from RTM 1.2;

- Susbsystem packages derived from railML 2.4 or railML 3.1:
    - Rolling stock takes over the properties described in railML 2.4, and adds an orientation to vehicles, so axles or pantographs can be designated sequentially without any ambiguity, over the whole train formation;
    - Track is derived from railML 3.1, and limits the component description to the most commonly encountered ones (running tracks, switches or crossings, buffers);
    - Energy is also reduced to the useful items, i.e. contact wire, and catenary characteristics incl. compatible pantograph specification.

The resulting “SIA” model shall not be construed as an “official release” of either RTM or railML. However:

- It demonstrates that tailoring a model for a definite purpose, using existing “building blocks”, is feasible and realistic, provided the models used have been designed for such purpose;
- It allows to test different ways of assembling existing UML models and generating data exchange formats;
- It provides material for further, more extensive modelling attempts (OPTIMA) and for a generalized “model federation” (Linx4rail).

See also section 3.6 regarding evolution of railML and RTM/RSM.

The coupling of RTM and railML packages was easy:

- railML diagrams show little dependency on RTM diagrams in the present context. These dependencies are:
    - inheritance from RSM class “NamedResource”, ensuring consistent naming and referencing: railML class “Rolling Stock”;
    - inheritance from RSM class “LocatedNetEntity”, ensuring the above, plus location of the net entity in arbitrary positioning systems: railML class ElectrificationSection, Track, and TrackNode;
    - reference to “standard” measurement values: classes “Length” and “Current” are referenced by railML;

Rolling stock positioning is provided via sensor positioning, which is part of the Observation & Measure package of RSM.

The overall resulting package diagram is as follows:

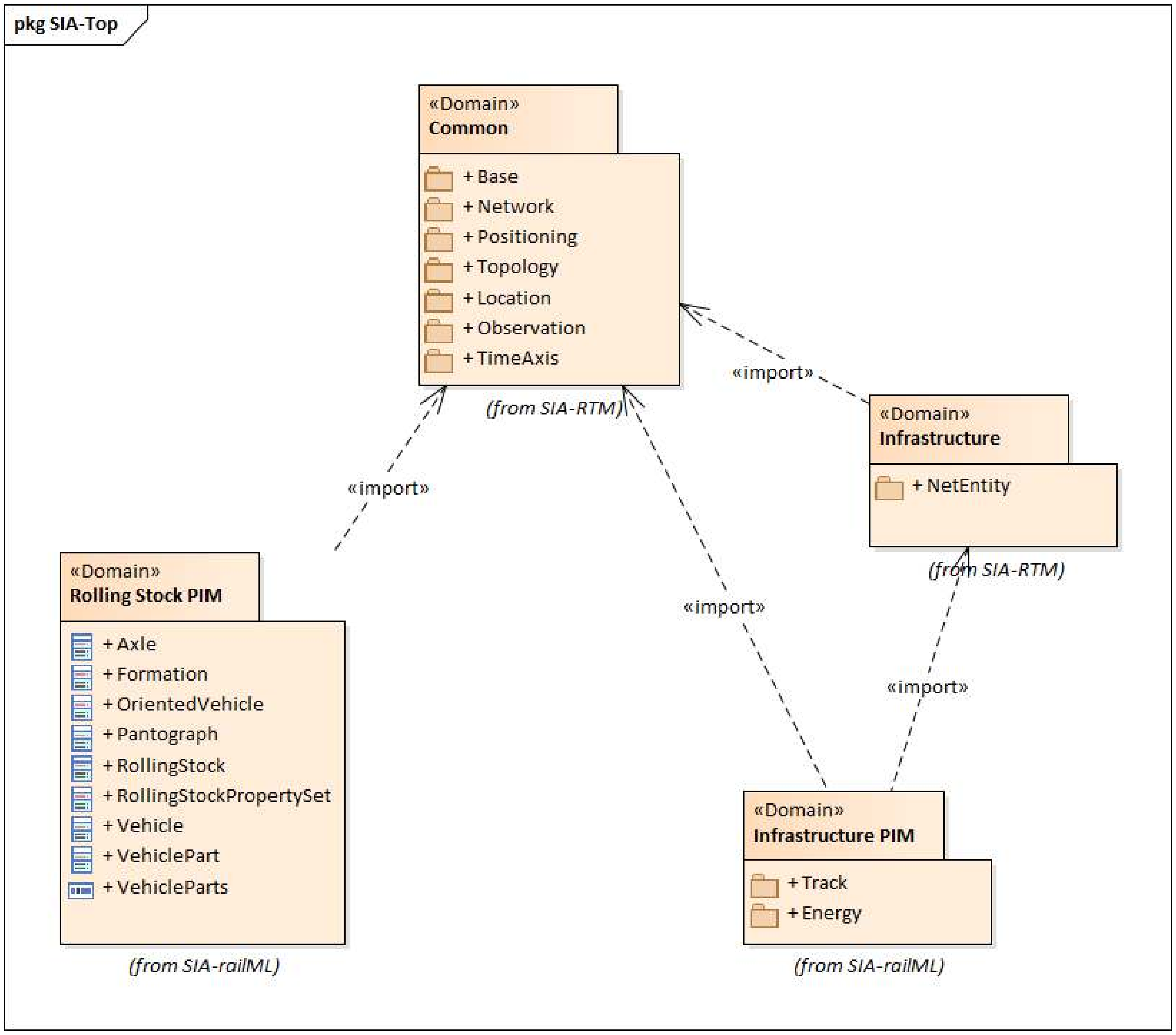


Figure 5: SIA Model

The two top packages are subsets of RSM 1.2, and the two bottom packages are subsets of railML. Dependencies are never bi-directional nor circular: following the “Dependency inversion” principle, itself the “D” in “SOLID” principles, abstractions do not depend on details, but details depend on abstractions: see for instance [32].

3.4 Deriving platform specific models

The UML modelling tool Enterprise Architect Corporate edition contains a set of tools for automatic generation of platform specific model implementations. This includes code generation for various programming languages as well as different domain models (see [33] and [34]):

- C# / C++
- Java

- PHP
- Python
- XSD

The focus of this section is on deriving a XML-based data model from the generic UML. There exist two ways for XSD generation. Both approaches are presented in the following sub-sections.

3.4.1 EA internal XSD generation

For generation of railML 3.1 scheme files the inbuilt XSD Generation facility of Enterprise Architect Corporate edition has been used. This procedure is described in detail in [35]. Here are the key facts:

- **Usage of UML Profile for XML**. The UML profile for XML specifies a set of stereotypes, which can be applied to a UML model in order to make it more suitable for a conversion to XSD. The stereotype explicitly tells the generator to which XSD structure the UML construct shall be mapped. For railML 3.1 the stereotypes *XSDcomplexType*, *XSDelement*, *XSDattribute*, *XSDsimpleType* and *XSDunion* have been used. The website [35] provides a detailed overview about the matching between the UML constructs and their XSD representation.

- **Usage of XSDDataTypes package**. The XSDDataTypes package provided by Enterprise Architect in form of a XMI file (see [36]) contains classes representing XSD primitive data types, e.g. xs:decimal.

- **XSD generation**. The generator is assessed by selecting the UML package and executing the XSD code generation in the toolbar menu as described in [37].

3.4.2 Post-EA XSD generation from XMI

OCTO developed an open-source tool uml2xsd, to create XML schema files (*.xsd) from XML Metadata Interchange (*.xmi) file. The advantage of this toolchain compared to the EA internal XSD generator is that it is completely under control w.r.t. its translating behaviour. The transformation is completely deterministic. Mapping principles are:

- UML packages are mapped to *.xsd files; package imports are translated into xsd imports;
- Each UML class is mapped to a top level XSD element and an XSD complex type;
- UML Inheritance is mapped to <xs:extension>;
- UML attributes are mapped to <xs:attribute> if of simple type, otherwise to <xs:element>;
- Attributes with lower value of cardinality = 0 are made “optional” in the XSD;
- Enumeration values are mapped as strings;
- Relations are represented by <xs:element>;

- The XSD IDREF mechanism is used to allow multiple relations to point to the same object instance in an XML document. In this case the "type" of the xs:element is replaced with tElementWithIDref.

While the input XMI is produced in the SPARX Enterprise Architect environment, the output XSD bears no specificity that would relate it to SPARX or other software. The output XSDs were checked for correctness using XML Spy, edited by Altova.

3.5 Proof of Concept

This section shall provide the proof of concept for the data modelling approach executed within the SIA project. It shall be shown that based on integrating the input models from railML 2.4, railML 3.1 and RTM on a platform-independent model (PIM) level, model adaptations can be done to fulfil SIA use case data modelling and data exchange requirements.

In particular, the focus is on generating XML schema based data exchange model from the PIM UML data model. For comparison of different methods to generate XSD files, both approaches described in the previous section have been run in parallel. The resulting XSD is an adaptation of the railML schema that can be considered as a proposal for schema evolution.

Step 1: Platform-independent modelling

PIM modelling in EA focuses on using only standard constructs from UML notation. In particular, classes, data types, enumerations and various forms of class relationships like association, generalization, composition and aggregation are being used. Apart from these standard UML constructs depicted in the left part of the following figure, EA tool box also offers XML schema specific constructs (right part). They have not been used in the context of this work.

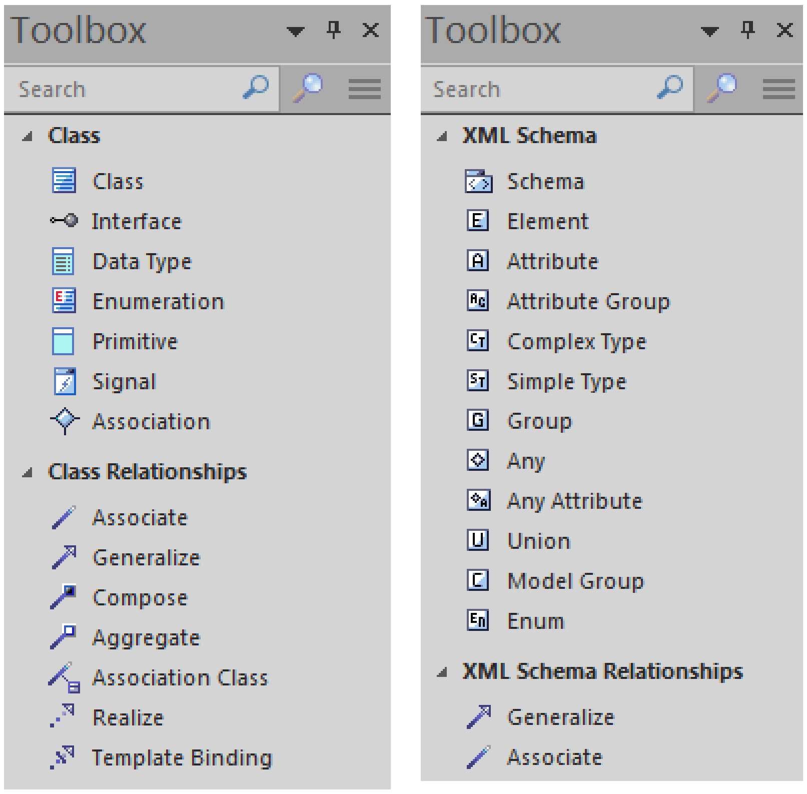


Figure 6: For PIM modelling only standard UML constructs (left) are being used

How does the PIM look like? For example, SIA data exchange use cases require modelling of track electrification. The following figure depicts the PIM UML model of track electrification:

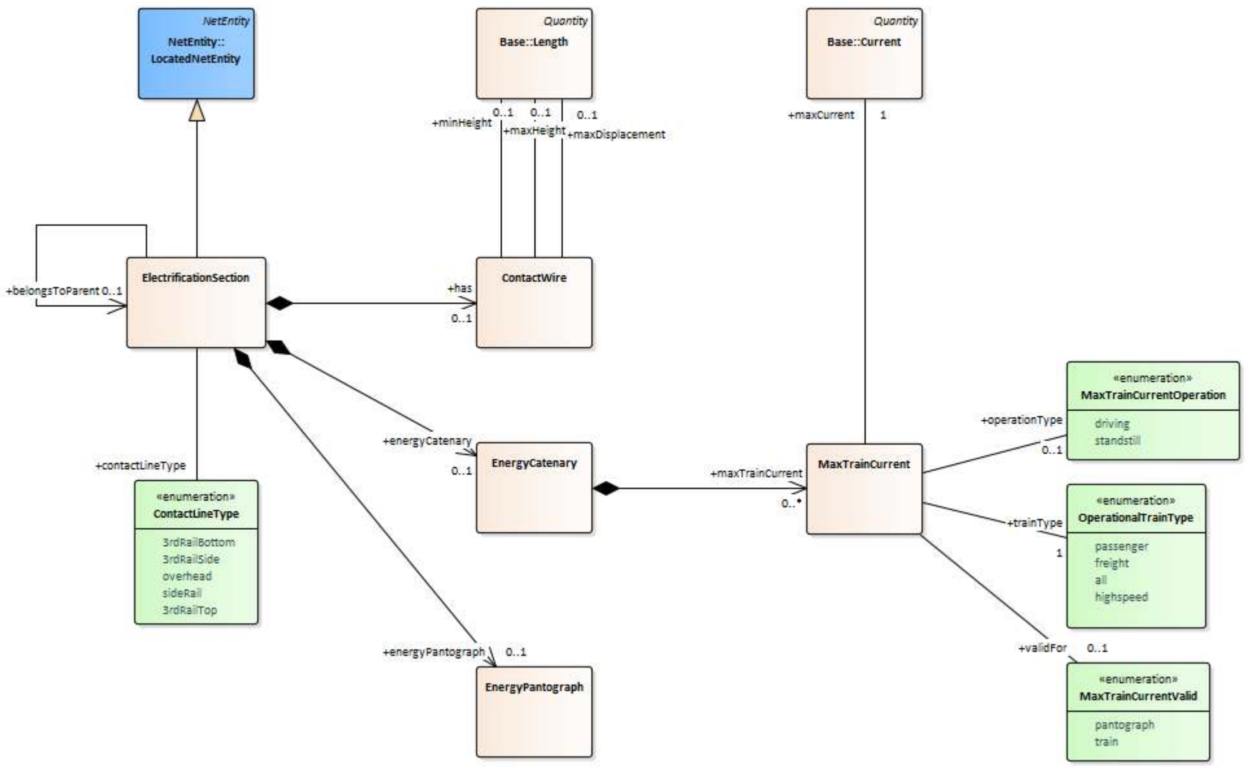
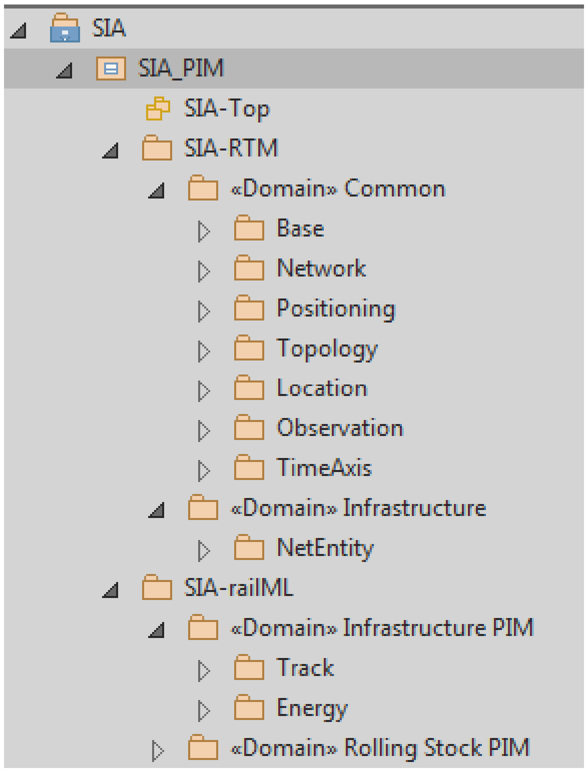


Figure 7: PIM UML model of track electrification

The PIM is structured like this:



- SIA
  - SIA_PIM
    - SIA-Top
    - SIA-RTM
      - «Domain» Common
        - Base
        - Network
        - Positioning
        - Topology
        - Location
        - Observation
        - TimeAxis
      - «Domain» Infrastructure
        - NetEntity
    - SIA-railML
      - «Domain» Infrastructure PIM
        - Track
        - Energy
      - «Domain» Rolling Stock PIM

Figure 8: EA Project Browser shows the structure of SIA PIM

As described in section 3.4 there are different ways how to get from the PIM UML to the PSM XSD. The following two sections describe the two most promising ones:

3.5.1 The "EA way"

<u>Step 2a: Transfer of PIM into PSM</u>

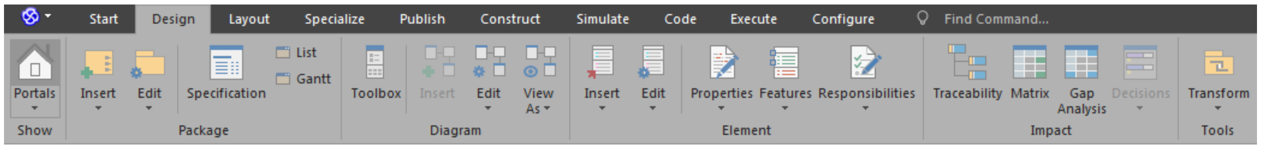
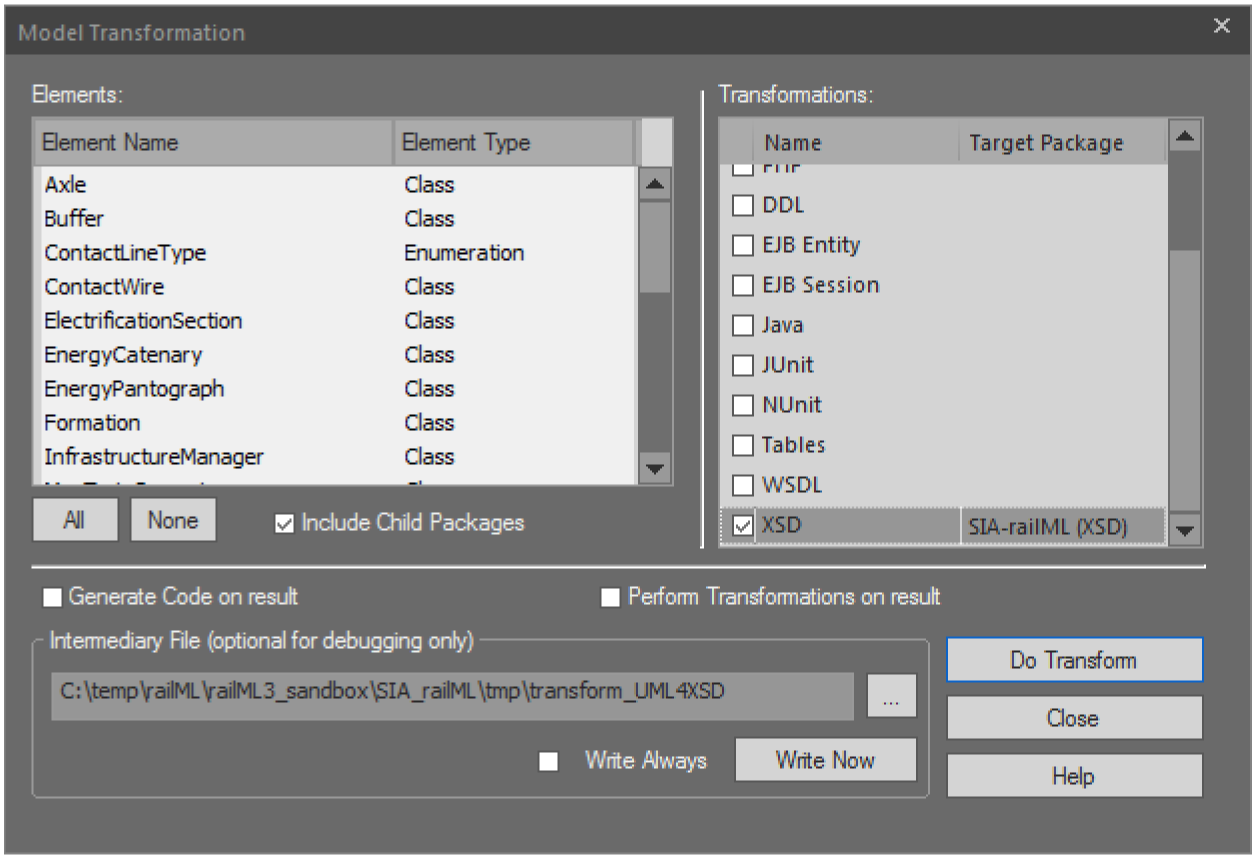


Figure 9: Model transformation tool is available in EA (menu Design)

Selecting the tool "Transform" in EA menu "Design" opens the following model transformation dialogue, where the target model "XSD" has been selected.



Model Transformation

Elements:

Element Name	Element Type
Axle	Class
Buffer	Class
ContactLineType	Enumeration
ContactWire	Class
ElectrificationSection	Class
EnergyCatenary	Class
EnergyPantograph	Class
Formation	Class
InfrastructureManager	Class

All | None | ☑ Include Child Packages

Transformations:

Name	Target Package
☐ DDL	
☐ EJB Entity	
☐ EJB Session	
☐ Java	
☐ JUnit	
☐ NUnit	
☐ Tables	
☐ WSDL	
☑ XSD	SIA-railML (XSD)

☐ Generate Code on result ☐ Perform Transformations on result

Intermediary File (optional for debugging only)

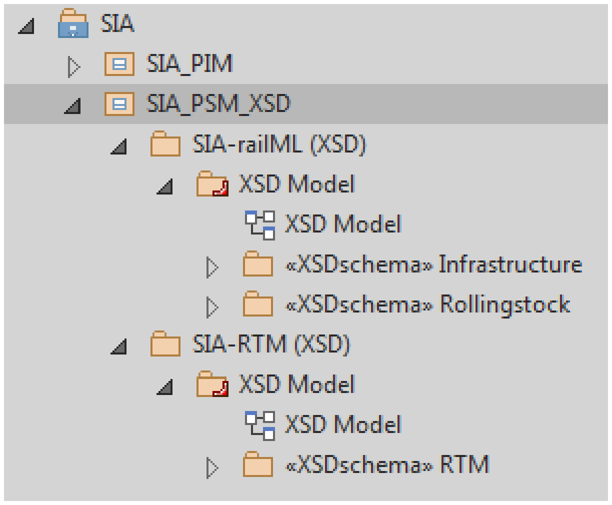
C:\temp\railML\railML3_sandbox\SIA_railML\tmp\transform_UML4XSD

☐ Write Always | Write Now

Do Transform | Close | Help

Figure 10: Generating PSM using EA inbuilt model transformation tool

As a result, a PSM has been generated. Every package of the PIM has been translated into a PSM package. The stereotype "*XSDschema*" indicates that every PSM package can be exported in a separate XML schema file.



```
SIA
  SIA_PIM
  SIA_PSM_XSD
    SIA-railML (XSD)
      XSD Model
        XSD Model
        «XSDschema» Infrastructure
        «XSDschema» Rollingstock
    SIA-RTM (XSD)
      XSD Model
        XSD Model
        «XSDschema» RTM
```

Figure 11: EA Project Browser shows the structure of the generated SIA PSM

For example, the Track electrification model introduced with Figure 6 looks like this after model transformation:

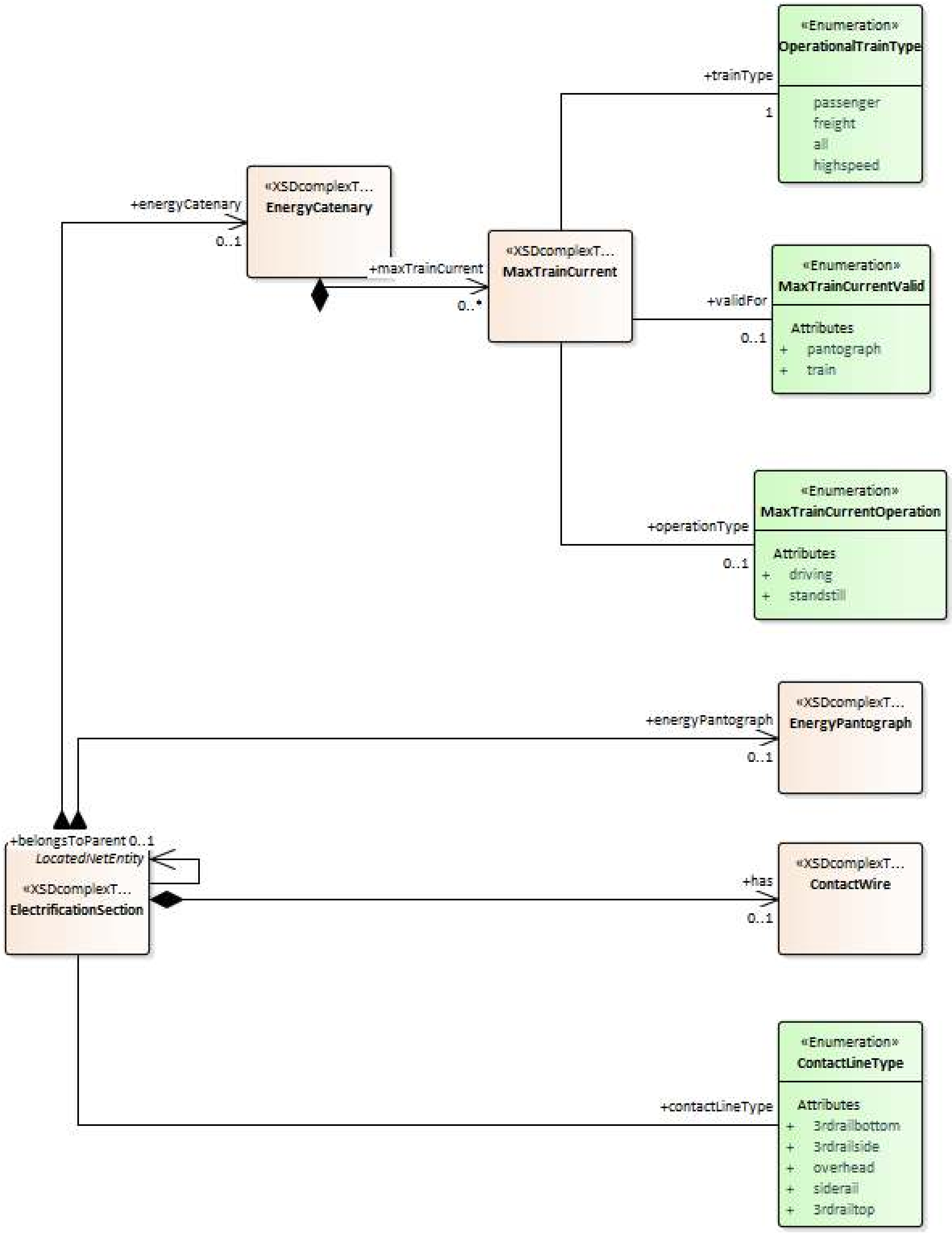


Figure 12: Generated PSM UML model of track electrification

Step 3a: XSD Code generation

The XSD oriented PSM provides the basis for model-driven generation of XML schema files as described in section 3.4.1.

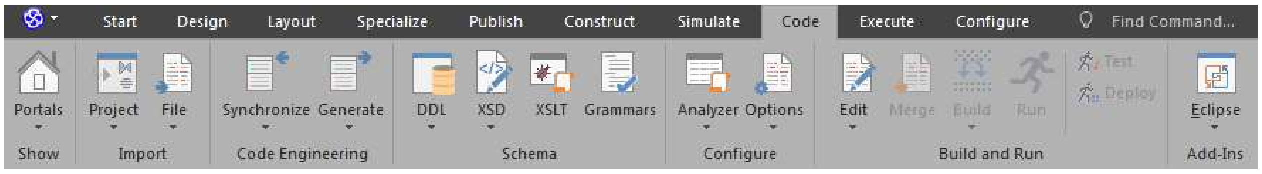
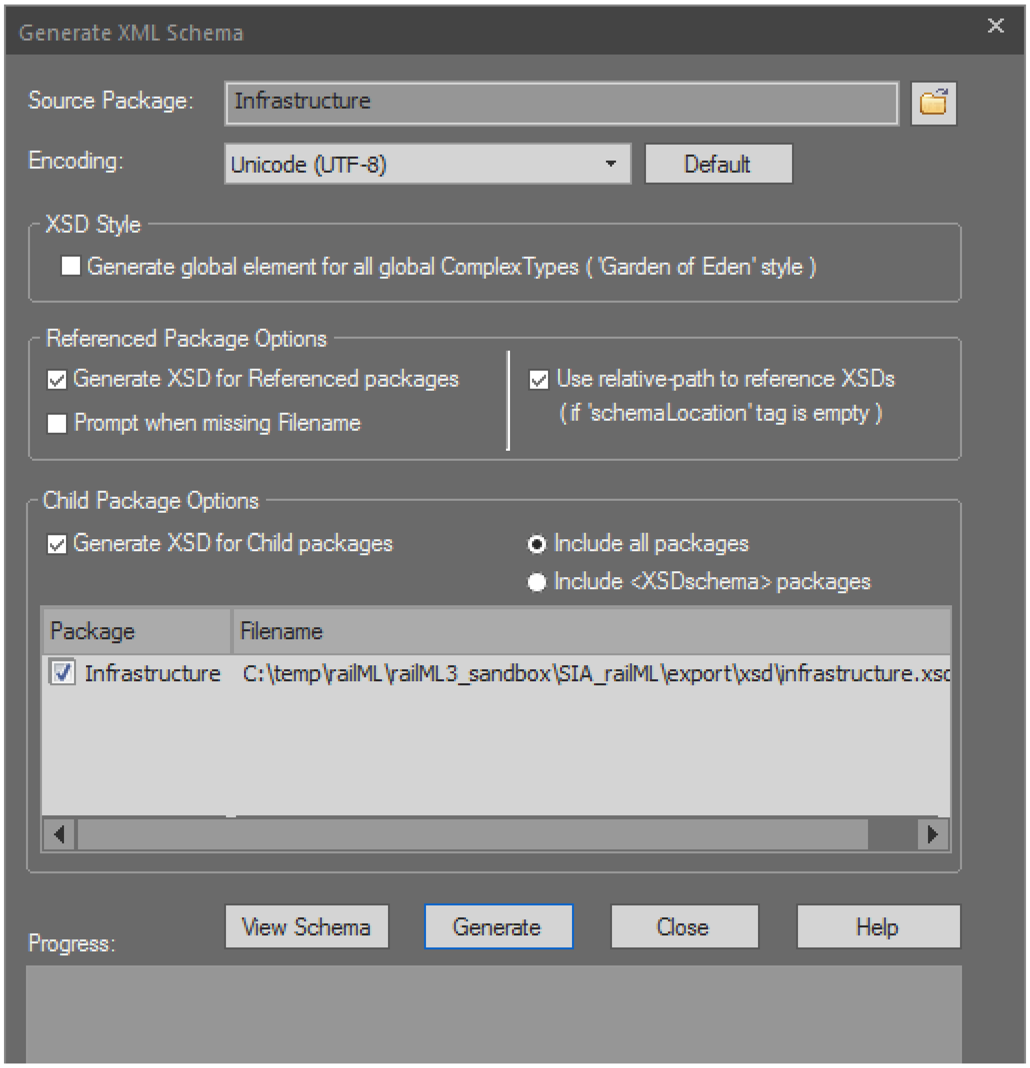


Figure 13: XSD file generation tool is available in EA (menu Code)

Browsing the tool box “XSD” in EA menu “Code” opens the following dialogue for generating XML schema file from a PSM UML model package.


Generate XML Schema

Source Package: Infrastructure

Encoding: Unicode (UTF-8) Default

XSD Style
- [ ] Generate global element for all global ComplexTypes ('Garden of Eden' style)

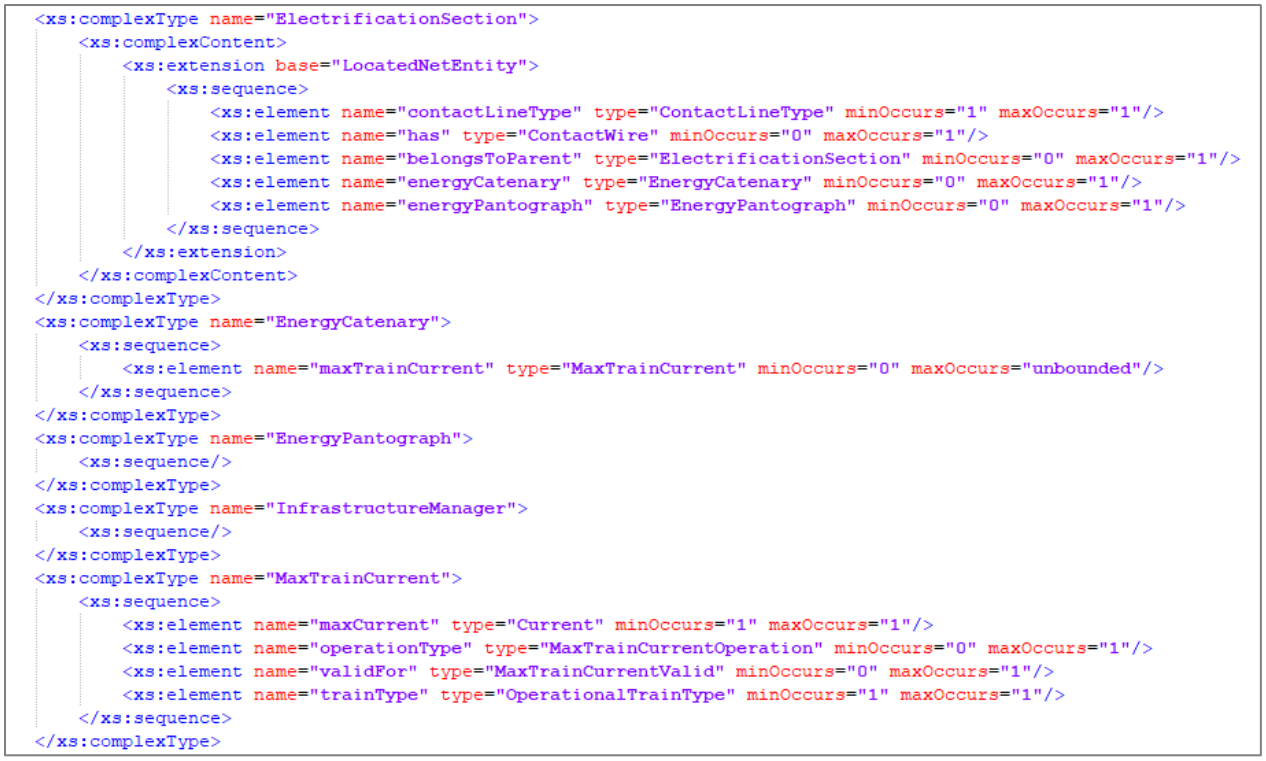
Referenced Package Options
- [✓] Generate XSD for Referenced packages
- [ ] Prompt when missing Filename
- [✓] Use relative-path to reference XSDs (if 'schemaLocation' tag is empty)

Child Package Options
- [✓] Generate XSD for Child packages
- [✓ Include all packages]
- Include <XSDschema> packages

Package	Filename
[✓] Infrastructure	C:\temp\railML\railML3_sandbox\SIA_railML\export\xsd\infrastructure.xsc

Figure 14: Exporting XSD using EA inbuilt XSD export tool

The following figure shows an excerpt from the generated XML schema file infrastructure.xsd. It includes the element *ElectrificationSection* that is already known from Figure 6 and Figure 11:



```
<xs:complexType name="ElectrificationSection">
    <xs:complexContent>
        <xs:extension base="LocatedNetEntity">
            <xs:sequence>
                <xs:element name="contactLineType" type="ContactLineType" minOccurs="1" maxOccurs="1"/>
                <xs:element name="has" type="ContactWire" minOccurs="0" maxOccurs="1"/>
                <xs:element name="belongsToParent" type="ElectrificationSection" minOccurs="0" maxOccurs="1"/>
                <xs:element name="energyCatenary" type="EnergyCatenary" minOccurs="0" maxOccurs="1"/>
                <xs:element name="energyPantograph" type="EnergyPantograph" minOccurs="0" maxOccurs="1"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="EnergyCatenary">
    <xs:sequence>
        <xs:element name="maxTrainCurrent" type="MaxTrainCurrent" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="EnergyPantograph">
    <xs:sequence/>
</xs:complexType>
<xs:complexType name="InfrastructureManager">
    <xs:sequence/>
</xs:complexType>
<xs:complexType name="MaxTrainCurrent">
    <xs:sequence>
        <xs:element name="maxCurrent" type="Current" minOccurs="1" maxOccurs="1"/>
        <xs:element name="operationType" type="MaxTrainCurrentOperation" minOccurs="0" maxOccurs="1"/>
        <xs:element name="validFor" type="MaxTrainCurrentValid" minOccurs="0" maxOccurs="1"/>
        <xs:element name="trainType" type="OperationalTrainType" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
```

Figure 15: Excerpt from generated XML schema file

3.5.2 The "XMI way"

<u>Step 2b: Exporting the model to XMI</u>

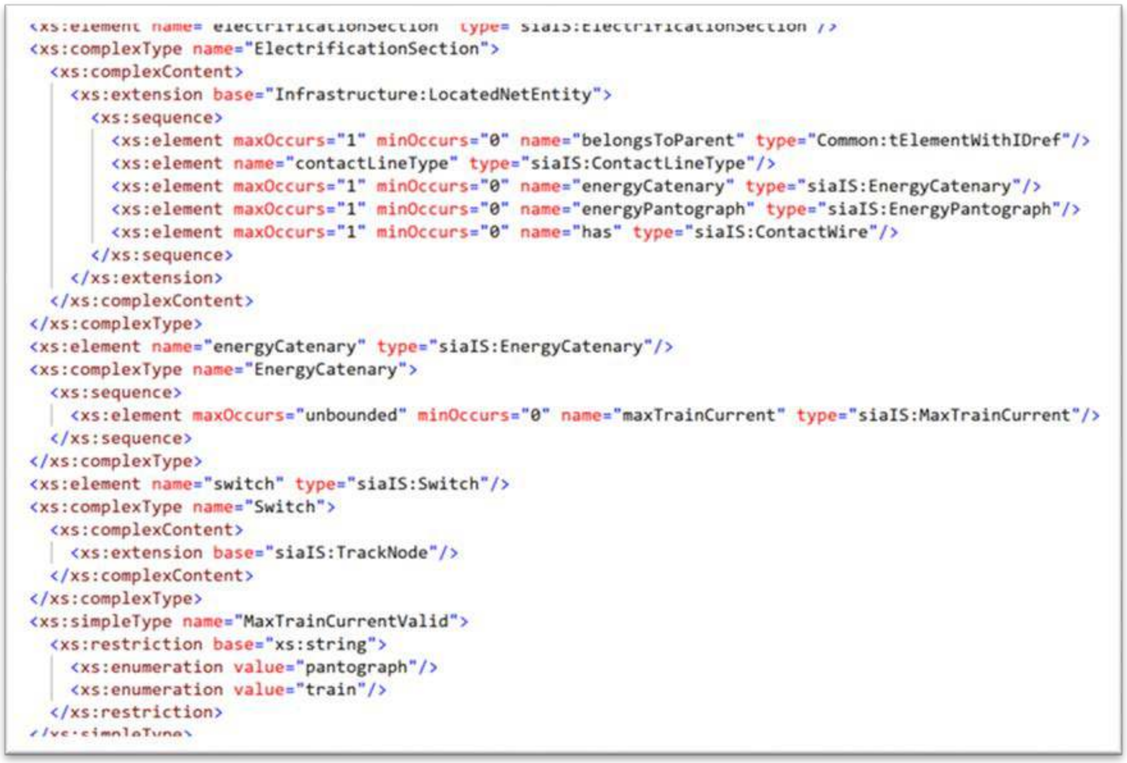
EA offers an inbuilt XMI export tool. Before the export, packages that shall be transferred into XML schema files in later stage are flagged with stereotypes "Domain".

<u>Step 3b: Running the UML2XSD tool</u>

The open source tool[6] developed by OCTO Technology, uml2xsd, is used to transform the XMI file into XML schema files. The mapping between the packages / elements / attributes / enumeration values is done as described in section 3.4.2.

The resulting XML schema files (one per "Domain"-stereotyped package) are exemplified by an excerpt shown below, matching the example previously given in Figure 14:

[6] The code is currently being shared between SNCF, UIC, EULYNX and other companies or organizations. Access to the source code is on request



```
<xs:element name="electrificationSection" type="siaIS:ElectrificationSection"/>
<xs:complexType name="ElectrificationSection">
  <xs:complexContent>
    <xs:extension base="Infrastructure:LocatedNetEntity">
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="0" name="belongsToParent" type="Common:tElementWithIDref"/>
        <xs:element name="contactLineType" type="siaIS:ContactLineType"/>
        <xs:element maxOccurs="1" minOccurs="0" name="energyCatenary" type="siaIS:EnergyCatenary"/>
        <xs:element maxOccurs="1" minOccurs="0" name="energyPantograph" type="siaIS:EnergyPantograph"/>
        <xs:element maxOccurs="1" minOccurs="0" name="has" type="siaIS:ContactWire"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:element name="energyCatenary" type="siaIS:EnergyCatenary"/>
<xs:complexType name="EnergyCatenary">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="maxTrainCurrent" type="siaIS:MaxTrainCurrent"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="switch" type="siaIS:Switch"/>
<xs:complexType name="Switch">
  <xs:complexContent>
    <xs:extension base="siaIS:TrackNode"/>
  </xs:complexContent>
</xs:complexType>
<xs:simpleType name="MaxTrainCurrentValid">
  <xs:restriction base="xs:string">
    <xs:enumeration value="pantograph"/>
    <xs:enumeration value="train"/>
  </xs:restriction>
</xs:simpleType>
```

Figure 16: Excerpt from XML Schema file generated by uml2xsd

There are a few differences with the previous example:

- Namespaces are systematically included as prefixes to names, preventing naming collisions from happening; here, both the local namespace (siaIS), other project namespaces (Infrastructure…) and the XML schema namespace (xs) are shown;
- When both maxOccurs and minOccurs are set to 1 in UML, they are omitted in the XSD and their default values (=1) apply;
- The ordering of elements and complex types, which is not part of the PIM, is different, as one would expect.

A more fundamental design option was chosen in uml2xsd, resulting in a significant difference from the previous generation channel: The XSD IDREF mechanism is used to allow multiple relations to point to the same object instance in an XML document.

It is used every time unless at least one of the following conditions is true:

1. The relation target is an enumeration
2. An EA tag with name "containment" and value "Value" is added on the XMI attribute
3. The relation is a composition
4. Relation targets a basic type (e.g. String)

In this case, the "type" of the xs:element will be replaced with tElementWithIDref. Obviously, when such referencing is not desirable, it can be avoided using the tags mentioned under item 2 above.

Step 4: Model Documentation

The last step, model documentation, obviously does not depend on the XSD production method.

Documentation of SIA model (elements, attributes etc.) is done on UML basis in EA. The documentation is then being exported using the EA inbuilt publishing tool. In particular, an HTML report is generated:

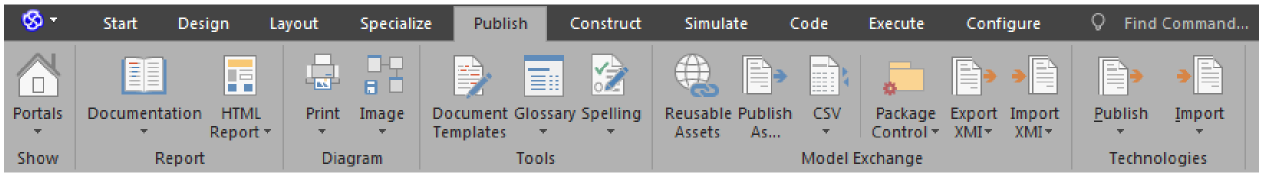
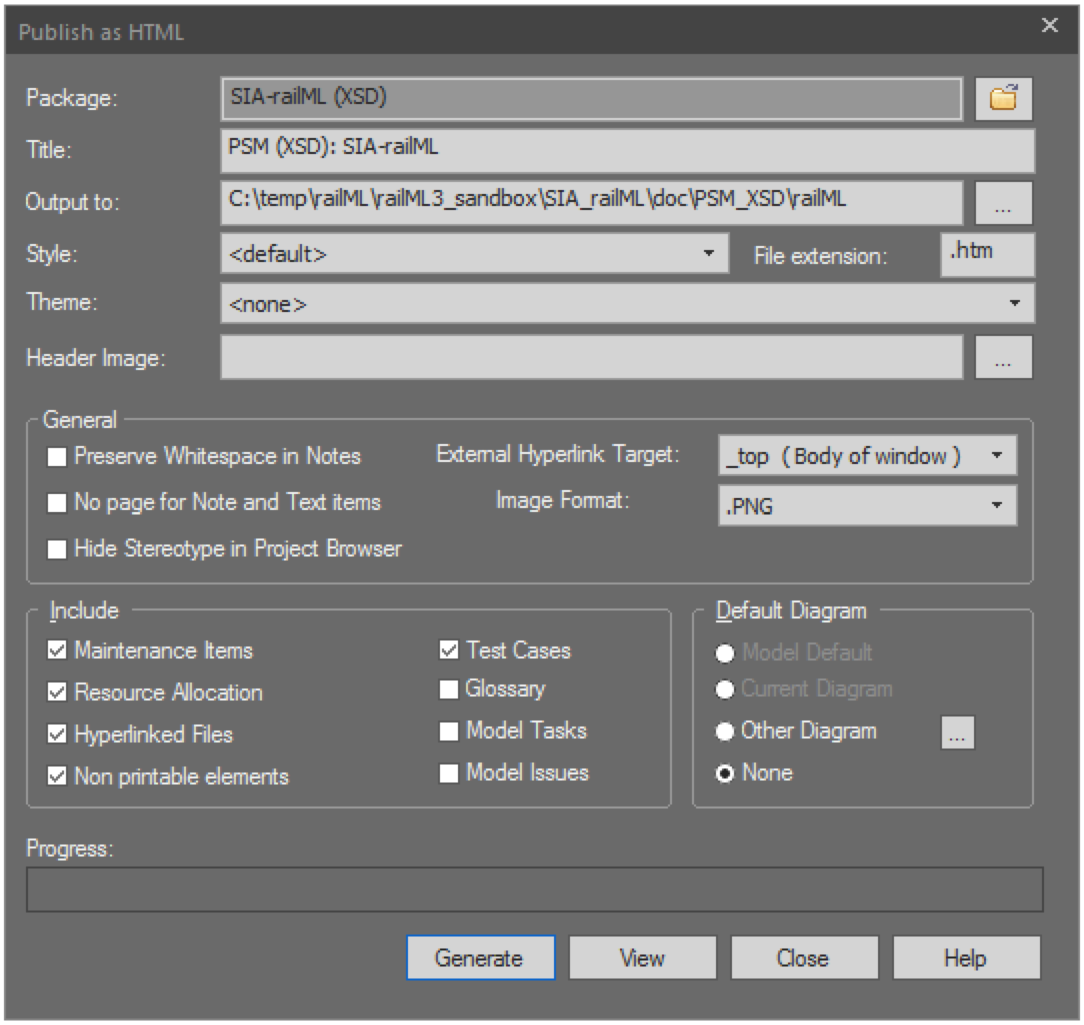


Figure 17: HTML report documentation is available in EA (menu Publish)

Choosing the tool “HTML report” in EA menu “Publish” opens the following dialogue:


Publish as HTML

Package: SIA-railML (XSD)
Title: PSM (XSD): SIA-railML
Output to: C:\temp\railML\railML3_sandbox\SIA_railML\doc\PSM_XSD\railML
Style: <default>
File extension: .htm
Theme: <none>
Header Image:

General
- ☐ Preserve Whitespace in Notes
- ☐ No page for Note and Text items
- ☐ Hide Stereotype in Project Browser

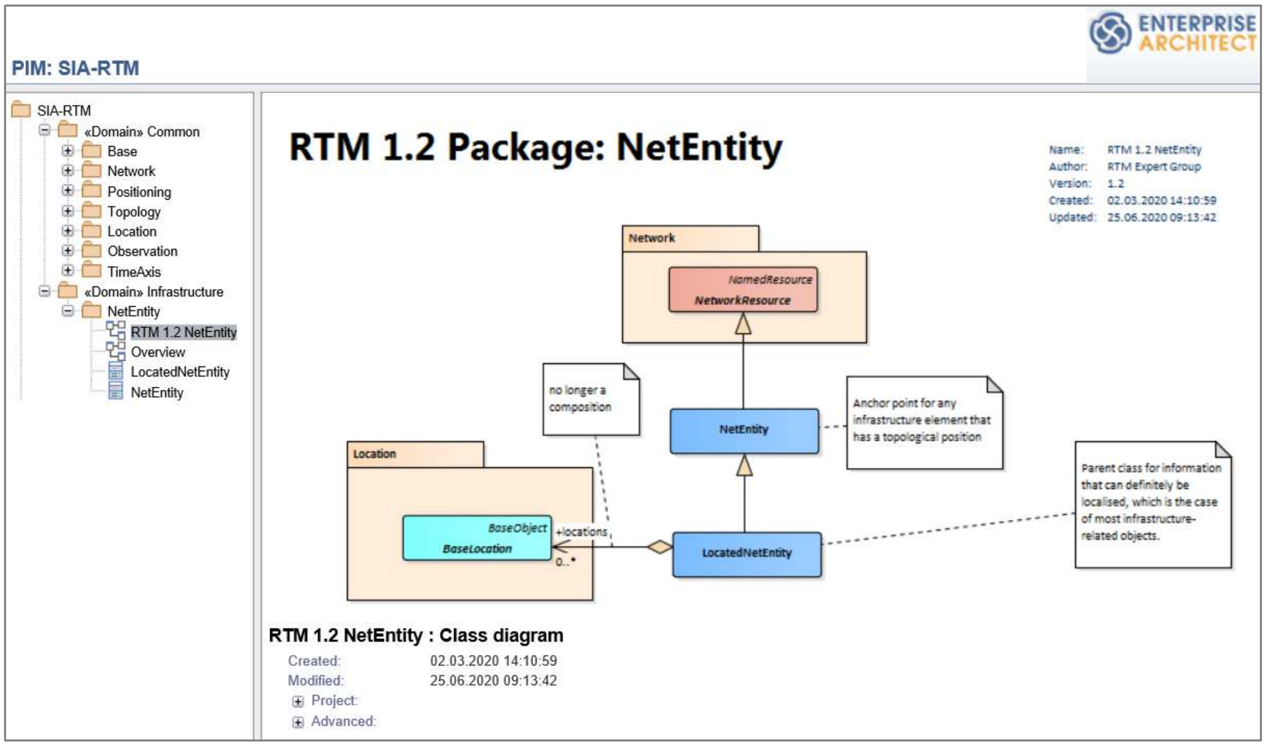
External Hyperlink Target: _top (Body of window)
Image Format: .PNG

Include
- ☑ Maintenance Items
- ☑ Resource Allocation
- ☑ Hyperlinked Files
- ☑ Non printable elements
- ☑ Test Cases
- ☐ Glossary
- ☐ Model Tasks
- ☐ Model Issues

Default Diagram
- Model Default
- Current Diagram
- Other Diagram
- [✓ None]

Progress:

Figure 18: Publishing UML model documentation using EA inbuilt of HTML report



RTM 1.2 Package: NetEntity

Name: RTM 1.2 NetEntity
Author: RTM Expert Group
Version: 1.2
Created: 02.03.2020 14:10:59
Updated: 25.06.2020 09:13:42

no longer a composition

Anchor point for any infrastructure element that has a topological position

Parent class for information that can definitely be localised, which is the case of most infrastructure-related objects.

RTM 1.2 NetEntity : Class diagram

Created: 02.03.2020 14:10:59
Modified: 25.06.2020 09:13:42

Figure 19: Excerpt from the HTML documentation report generated by EA

3.6 Impact on RTM and railML development

3.6.1 railML

The development of railML follows a community approach as described in section 4.2.2. This means that the railML schema can be adapted upon request if there is an interest by at least three parties, and – in case of railML 3.x – if there exists a specific use case.

The SIA project addresses several different data exchange use cases. Some of them have already been formulated as railML use cases public available on the railML wiki:

- Positioning and Map-Matching (see [22]])
- Asset status representation (see [23])

Depending on their priority among the partners of the railML community, these use cases will be implemented with one of the future railML versions. Other data exchange use cases mentioned in this report, have not been communicated to railML.org, yet:

- Maintenance procedures and recommendations
- Auscultation data
- Inspection data

Consequently, there remain objects, which have not yet been requested to be implemented in the railML schema. It is an open task for the SIA project partners to communicate their required data exchange use cases towards the railML community in order to trigger a model implementation / adaptation with future versions of the railway data exchange standard.

The XSD generated within this project WP is an adaptation of the railML schema to be understood as a proposal for schema evolution. Due to the community based development approach described before it cannot be guaranteed, that future railML schema versions will follow this proposal in detail. However, by communicating the model adaptations forming the basis for the modified XSD towards the railML community, it is ensured that the required changes can be discussed with other users and stakeholders of the sector leading towards a solid model solution.

3.6.2 RSM

RSM evolution is driven by following principles:

- **The RSM model is platform-independent and usage-agnostic**. This means that particular use cases do not entail modifications of the RSM packages, unless of course such use cases would reveal an insufficient abstraction level in the RSM model. Specialized models (such as EULYNX for signalling) are expected to use some RSM primary concepts, but there shall be no dependencies the other way round;
- **The RSM model can be used “standalone” for certain usages**. Examples are geographic and geometric description of railway networks, or rolling stock running simulation. However RSM avoids accumulation of detailed objects or properties that are better managed by one specialized model, and are not used by many others.
- **The RSM model allows the coupling of multiple, specialized models**. RSM provides the basic “vocabulary elements” that other models are likely to need, such as positioning (location and time) in multiple referencing systems, or measurements (values, units, circumstances). For such purpose, RSM rests on well-established de facto standards published by the W3C or the OGC.

Particular projects, such as SIA, allow to test the capabilities already provided. In the SIA case, the “Time Axis” and “Observation & Measure” packages are under particular scrutiny.

4 Conclusion and Outlook

The aim of the SIA D5.3 deliverable was to adapt existing, proven industry standards (RTM / RSM, railML) to the SIA use cases.

The genericity and extensibility of the aforementioned models and XML schema definitions is adequate. The effort to adapt to the use cases can be called minimal, and this is due

- To the abstraction of the pre-existing models;
- To strict separation of concerns between packages, and between classes within packages;
- To the genericity of the XSD production chains.

While the used models are platform-independent, one particular, proprietary modelling tool (SPARX Enterprise Architect, mixing versions 11 and 14) was used out of convenience. This choice does not preclude other choices to be made by other users, keeping in mind that XMI, as a model exchange format, is not well standardized[7], so adjustments may be needed.

Moreover, XSD generation can be performed in different, equally valuable ways, and does not require proprietary software.

Further adaptation to the SIA use cases would best be done in the UML models, as UML (being a graphic language) provides the best way to perform adaptations without bloating or other unwanted side-effects. Also, the UML to XSD mapping principles need not be revised, and will ensure that updated XSDs will not greatly differ from previous ones, thus requiring minimal adaptations from client software.

The XSD generated within this project WP is an adaptation of the railML 3.x schema to be understood as a proposal for schema evolution. Due to the community based development approach described in this report, it cannot be guaranteed, that future railML schema versions will follow this proposal in detail. Therefore, SIA project needs to communicate the model adaptations forming the basis for the modified XSD towards the railML community (e.g. in form of forum postings), in order to ensure that the required changes can be discussed with other users and stakeholders of the sector leading towards a solid model industry standard solution.

[7] Note that the authors routinely use the open source Eclipse environment that could be considered an alternative. Eclipse does not however offer the same rich publishing tools as EA.

References

[1] buildingSMART International: *Industry Foundation Classes: extensions for rail*: https://www.buildingsmart.org/ifc-rail-candidate-standard-is-available-for-review-and-comment/

[2] EULYNX: *EULYNX Data preparation model*: https://www.eulynx.eu/index.php/dataprep

[3] EULYNX: *EULYNX DataPrep public snapshot*: https://dataprep.eulynx.eu/index.htm

[4] European Commission: *(Cordis) Intelligent Innovative Smart Maintenance of Assets by integRated Technologies*; https://cordis.europa.eu/project/id/730569; last access: 20.05.2020

[5] European Commission: *(Cordis) Innovative Intelligent Rail*; https://cordis.europa.eu/project/id/635900; last access: 20.05.2020

[6] In2Rail: *Innovative Intelligent Rail*. http://www.in2rail.eu/; last access: 20.05.2020

[7] In2Rail *Project Deliverable D9.1: "Asset status representation"*; October 2016.

[8] IN2SMART *Project Deliverable D7.1: "Open data: a review of the state-of-the-art"*; August 2017.

[9] IN2SMART *Project Deliverable D7.2: "Requirements analysis and definition of standard guideline"*; August 2018.

[10] Karlsson, M. and Wegele, S. (2017): *Digitalisation - Toward seamless connectivitiy*; in: Global Railway Review, Vol. 23, issue 05.

[11] Nash, A.; Huerlimann, D.; Schütte, J.; Krauss, V.P. (2004): *RailML – a standard data interface for railroad applications*. In: Computers in Railways IX, pp. 233-240

[12] railML.org: *Official railML website*. https://www.railml.org/en/; last access: 14.05.2020

[13] railML.org: *The railML subschemas*. https://www.railml.org/en/user/subschemes.html; last access: 14.05.2020

[14] railML.org: *railVIVID – The railML Viewer & Validator powered by UIC*. https://www.railml.org/en/user/railvivid.html; last access: 14.05.2020

[15] railML.org: *railML® partners*. https://www.railml.org/en/introduction/partners.html; last access: 14.05.2020

[16] railML.org: *DLR Institut für Verkehrssystemtechnik*. https://www.railml.org/en/introduction/partners/detail/dlr-braunschweig.html; last access: 14.05.2020

[17] railML.org: *Licence terms*. https://www.railml.org/en/user/licence.html; last access: 14.05.2020

[18] railML.org: *Certification of your railML® interface*. https://www.railml.org/en/developer/certification.html; last access: 14.05.2020

[19] railML.org: *railML Forum*. https://forum.railml.org/; last access: 14.05.2020

[20] railML.org: *railML Wiki Version 2*: https://wiki2.railml.org/index.php?title=Main_Page; *railML Wiki Version 3*: https://wiki3.railml.org/index.php?title=Main_Page; last access: 14.05.2020

[21] railML.org: *(Wiki) UC: Use Cases*. https://wiki3.railml.org/index.php?title=UC:Use_cases; last access: 14.05.2020

[22] railML.org: *(Wiki) UC:IS:PositioningAndMap-Matching.* https://wiki3.railml.org/index.php?title=UC:IS:PositioningAndMap-Matching; last access: 17.06.2020

[23] railML.org: *(Wiki) UC:IS:Asset status representation*. https://wiki3.railml.org/index.php?title=UC:IS:Asset_status_representation; last access: 17.06.2020

[24] railML.org: *railML Trac ticket system*. https://trac.railml.org/; last access: 14.05.2020

[25] railML.org: *railML Subversion repository*. https://svn.railml.org/; last access: 14.05.2020

[26] Shift2Rail: *LinX4Rail – System architecture and Conceptual Data Model for rail, a S²R project*; see https://projects.shift2rail.org/s2r_ipx_n.aspx?p=LINX4RAIL

[27] SIA Project *Deliverable D2.1: "End user requirements of SIA and validation plan"*

[28] SIA Project *Deliverable D2.2: "SIA Architecture and Verification Plan"*

[29] SIA Project *Deliverable D3.2: "Development of railway environment specific positioning algorithm"*

[30] SIA Project *Deliverable D4.1: "Wheel to rail and pantograph to catenary sensing nodes"*

[31] SIA Project *Deliverable D4.3: "On-board data integration platform and train-track communication hub"*

[32] *SOLID principles* (applying to object-oriented design and coding), see https://codedesignetc.com/2017/03/09/solid-principles-in-oops/

[33] SparxSystems: *Enterprise Architect – Generate Source Code*; https://sparxsystems.com.au/enterprise_architect_user_guide/15.1/model_domains/generatesourcecode.html; last access: 17.06.2020

[34] SparxSystems: *Enterprise Architect – Schema Composer Profiles*; https://sparxsystems.com/enterprise_architect_user_guide/15.1/model_domains/schema_composer_profiles.html; last access: 24.06.2020

[35] SparxSystems: *Enterprise Architect – XML Schema Generation*; https://www.sparxsystems.com/resources/xml_schema_generation.html; last access: 04.06.2020

[36] SparxSystems: *Enterprise Architect – XSDDataTypes package XMI*; https://www.sparxsystems.com/downloads/profiles/XSDDataTypes.xml; last access: 04.06.2020

[37] SparxSystems: *Enterprise Architect – Generate XSD*; https://sparxsystems.com/enterprise_architect_user_guide/15.1/model_domains/generate_xsd.html; last access: 17.06.2020

[38] UIC: *RailTopoModel (RTM) version 1.1 release webpage*: https://uic.org/rail-system/railtopomodel

[39] Wikipedia: *railML.* https://en.wikipedia.org/wiki/RailML; last access: 17.06.2020