Formalization and Processing of Data Requirements for the Development of Next Generation Railway Traffic Management Systems*

Airy Magnien[1][0000−0003−3583−2029], Gabriele Cecchetti[2][0000−0002−0910−6739], Anna Lina Ruscelli[2][0000−0003−1995−048X], Paul Hyde[3][0000−0001−5817−0107], Jin Liu[3][0000−0002−3808−5957], and Stefan Wegele[4]

[1] SNCF and UIC, France airy.magnien@sncf.fr
[2] Scuola Superiore Sant'Anna, Italy {g.cecchetti,a.ruscelli}@santannapisa.it
[3] School of Engineering, Newcastle University, UK {paul.hyde, jin.liu}@newcastle.ac.uk
[4] Siemens, Germany stefan.wegele@siemens.com

Abstract. Railway Traffic Management Systems (TMSs) handle data from multiple railway subsystems, including Rail Business Services (such as interlocking, RBC, maintenance service, etc.) and external services (such as passenger information systems, weather forecast, etc.). In turn, the data from these subsystems are described in several models or ontologies contributed by various organizations or projects which are in a *process* of converging or federation. The challenge of the Shift2Rail OPTIMA project, which is implementing a communication platform for virtual testing of new applications for railway TMS, is to allow the exchange of data between different services or users and to support new traffic management applications, enabling access to a large number of disparate data sources. In this paper, the core activities of the OPTIMA project related to the formulation and standardization of a common data model are described. A new Common Data Model is developed based on standardized data structures to enable the seamless exchange of large amounts of data between different and heterogeneous sources and consumers of data, that contributes to the building of next generation of a more effective and efficient railway TMS suitable to offer precise and real-time traffic information to railway operators and other end users.

Keywords: Railway Traffic Management System, Common Data Model, model transformation, context-free grammar.

1 Introduction

Railway Traffic Management Systems (TMSs) are used in Operation Control Centers to monitor and manage Railway Business Services like traffic management, maintenance and energy systems, and external services like the Passenger

* Supported by H2020 Shift2Rail OPTIMA project G.A. 881777

Information Systems. However, since current legacy TMSs are lacking in standardized communication interfaces with internal and external services and in interoperable data structures, the interoperability with different TMSs and the upgrade of these systems is difficult. New generation Traffic Management Systems aim to overcome these limitations and take advantage of the capability to access multiple disparate data sources in order to better optimize operational solutions, as well as increase the integration between system.

The European Commission promotes the design and implementation of innovative solutions suitable to outline an European railway transport mode more competitive, sustainable, interoperable, and efficient through the Shift2Rail (S2R) Joint Undertaking, set under the H2020 program that aims to specify and design railways systems based on standardised frameworks and a Common Data Model (CDM).

2 OPTIMA project

One of the challenges of TD2.9 of Innovation Programme 2 in S2R Multi-Annual Action Plan [1, 2] is to develop a Technological Demonstrator providing seamless data exchange to support future TMSs which enables the integration of status information from different services. The H2020 S2R OPTIMA (cOmmunication Platform for TraffIc ManAgement demonstrator) project [7], started in December of 2019, is strictly linked to TD2.9, it aims to implement and validate a demonstrator of a communication platform for the testing and validation of novel industry solutions for next generation TMSs. The components of the communication platform developed by OPTIMA are conceived to ensure seamless access to persistent data from heterogeneous data sources with automated data exchange process, real time availability and configurable quality of service levels for services [3, 6]. The Integration Layer (IL) is the core component providing the functionalities of a middleware between the sources and consumers of data, and Traffic Control and Management Applications, hosted in the Applications Framework that provides a uniform deployment environment in which to deploy various TMS services into virtual machines or containers with a plug-and-play approach. The seamless exchange of data is ensured by the use of standardized and interoperable data structures and processes based on the definition of a CDM. Finally, in the Operations Center, newly designed Operator Workstations (OWs) enable operators to access the available data in IL via associated TMS applications. The OPTIMA demonstrator will be validated by connecting external prototypes from the S2R complementary projects X2RAIL-4 [9] and FINE-2 [4].

3 Railway system modelling: state of play

3.1 Shared models

LinX4Rail [5], an ongoing Shift2Rail project, aims at the federation, and ultimately, convergence of railway system models into a "Common Data Model".

LinX4Rail developments rest on models that are provided by various entities for different purposes. Model convergence requires, inter alia:

- harmonized semantics (Work Package 2), using ontologies;
- linking of models generally published in UML (Work Package 3);
- formalizing a shared railway system architecture and dealing with its governance (Work Package 5).

The so-called "candidate models", expected to become parts of the model federation, are currently:

Model	Purpose	Owner
RSM	multi-purpose rail system model	UIC
EULYNX DataPrep	Signalling assets (material or immaterial)	EULYNX
IFC Rail	Railway infrastructure assets (all subsystems, mostly material)	buildingSMART Intl.
TRANSMODEL	Multi-modal passenger traffic management and related assets	CEN
X2RAIL-4	Data exchange model for operational purposes, incl. ATO	X2RAIL-4 consortium

All the models above are shared between project participants at least. With the exception of X2RAIL-4, all have been published by their owners at different stages of completion, using various licenses.

3.2 Platform-specific model: X2Rail-4

Amongst the models listed above, the X2RAIL-4 model has a special status due to its distinct purpose and the formats adopted (JSON and Protobuf). The X2Rail4-model was developed during several Shift2Rail projects dedicated to TMSs starting in 2015 and reached its current version in the X2Rail-4 project, therefore the model name. The TMS as a central controlling instance requires data from almost all railway domains, including the infrastructure, interlocking, energy system, timetables, etc. To allow evolutionary extension of TMS with new functionalities, a common data architecture (data model) and common communication architecture (Application Programming Interface (API) for data access) were specified. This communication platform is used as a common backbone for several demonstrators developed by the complementary projects including Automatic Train Operation (ATO), Decision Support System, Connected Driver Advisory System, etc. [9]. The data model developed in X2Rail-4 is specially adapted for usage in this X2Rail-4-communication platform (called Integration Layer). It supports two serialisation formats, namely a human-readable format (JSON) and a binary format (Protobuf), which can be used interchangeably via API.

4 OPTIMA: from requirements to model

The initial intent of OPTIMA project was to rely to the "Common Data Model" prepared by Linx4Rail as the basis for deriving a platform-specific model, to be used by the IL. However, calendar constraints made such derivation difficult. Moreover, software applicable to the tasks had already been developed, using models prepared under other closed or running Shift2Rail projects. A pragmatic decision was made, which is to use the X2RAIL-4 model:

- the link between OPTIMA and the LinX4Rail CDM is preserved, even though indirectly (via X2Rail-4);
- the existing developments are preserved, but
- X2Rail-4 model evolution requires close cooperation between owner and user (here, OPTIMA) without creating undue dependencies.

While a cooperation agreement set the scene for that cooperation to happen, technical challenges remained.

4.1 Challenges of model evolution

The X2Rail-4 model is the basis for application developments using the IL. Such applications have been developed (implying backward compatibility of model changes) or are under development (implying openness to changes) in different projects, OPTIMA being one of them. The challenge is therefore to become able to extend an evolving model with requirements that are themselves evolving, avoiding cross-dependencies to the largest extent possible.

4.2 Previous works

Model-to-Model transformation was extensively explored, requirements-to-model, much less so. The authors acknowledge that published solutions have potentially been overlooked. The most commonly investigated model-to-model transformation path is from natural text requirements to UML. The absence of any well-established, shared terminologies or ontologies in the railway field that is precisely defined, documented, and used, is a significant issue in terms of establishing data models.

5 Formalizing data requirements

In general, data requirements are formulated in human-readable documents, and OPTIMA is no exception. Usage of requirement management tools is still uncommon, in international railway projects, although some project partners may be familiar with such tools. In this regard, the railway world does not seem to belong to the 17.8% of survey respondents having "strong knowledge" of using requirements management tools, but definitely to the 69% using a "systematic methodology" for collecting requirements [8].

Natural Language Processing can be excluded from the solutions, lacking comprehensive, published, and widely used domain ontologies[5]. As a matter of fact, Ontorail.org is the place where the ontology extractions from Linx4Rail candidate models are assembled, but the linkage of the extracted ontologies has just started.

Experience with some modelling endeavours in international projects (such as IFC Rail) showed that the set up of UML models would include two phases:

- domain requirement expression by domain experts, usually organized as collections of tables (with columns "objects", "description", "illustration"...);
- UML formalization by tandems, or teams, grouping domain experts and UML modelling experts.

Domain representation is time-consuming mainly because of the temptation of being complete and the resolution of overlaps, to ensure consistency and non-redundancy of the complete set of requirements.

UML formalization is time-consuming because it requires participants to understand some of the expertise or concerns of the other side (domain vs. modelling), which is no small effort, and because modelling choices have to be made. Additionally some domain knowledge is based on accepted practice and historical conventions, therefore certain domain concepts are sometimes expressed in requirements with contextual assumptions as to the meaning of terms.

Our goal was to find a *deterministic* solution to a *simplified* problem:

- requirements should be expressed in a formal, prescribed way, also dealing with semantic uncertainties;
- requirements should extend an existing model, not alter it;
- all transformations are done by code, using requirements and transformation options persisted in text files.

5.1 Minimal requirements... for requirements

Domain experts would spontaneously describe material or immaterial assets as "systems" composed of "objects" having "features" or "properties", a description that naturally evokes object-oriented modelling. Such views however collide with another valid world representation that would consider data exchange, resulting in bundling "data" into nested "data sets", i.e. coherent pieces of knowledge for a purpose.

In our case, we would only expect the domain expert to identify single objects and single object features, which is our "atomic" level. Features are defined and described in the context of the object they characterize, which is restrictive: ontology properties for instance are classes, and LinkML allows to define "slots" (equivalent to our "features") to be defined separately from classes and shared by several classes[6].

[5] ifcOWL is one such initiative, but extension of ifcOWL to the scope of IFC Rail is pending. See https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/

[6] Link Modeling Language, see https://github.com/linkml/linkml

This restriction is certainly old-fashioned, but simplifies the expression of requirements, at the cost of possible repetitions. Object features may be understood as attributes (or fields in a document), or references to other objects (or documents).

Object	Feature	Info	Feature type	Units or values	Priority	Source	Static	Quasi-static	Dynamic	PSM module	Responsible PSM class	Feature name in PSM
Track										TP	TrackView	
Track	ID	The ID of the Track	**unique_id**		1	D5.1-P53	Y			TP	TrackView	id
Track	DataValid	Data logged in the files are, by definition, valid data so this field is always YES	**boolean**		1	D5.1-P53		Y		TP	TrackView	?
Track	Occupance	occupation status of the track	**boolean**		1	D5.1-P53			Y	TP	TrackView	?
Track	Trains	list of trains occupying the track	**list of *Train**		1	D5.1-P53			Y	TP	TrackView	?
Track	Direction	Default moving direction	**enum**	**default_direction_01, default_direction_10**	2	D5.1-P53		Y		TP	TrackView	?
Track	DirectionChange	Change of the default direction	**boolean**		2	D5.1-P53		Y		TP	TrackView	?

Fig. 1. Requirements sheet (excerpt)

Using spreadsheets for input is common practice, and unlikely to deter domain experts. An excerpt of the used spreadsheet is shown in Fig. 1. Each requirement is self-contained, and is expressed in a single row. While the actual requirements viewing and editing environment is MS Excel, the work does not require more than CSV capabilities. System views are outside the scope of the formalisation approach adopted, however, it does not preclude them:

- users may use multiple sheets and files to sort requirements, but the processor will ignore this sorting;
- by design, model extension will not break the input model structure and will respect the system (or documentation) breakdown that was initially intended.

5.2 Supporting grammar

Some consideration was given to the data structure used to capture the requirements, particularly the grammar used, for instance, increasing the number of columns to accommodate finely tuned requirements would lead to "sparse matrices", the kind that is not easily edited, let alone reviewed. For example, an enumerated feature has a list of values, while a numeric feature has a unit, so two filled columns should be able to express both cases, instead of four columns with two irrelevant ones.

A pragmatic solution consisted of setting up a short context-free grammar, formalized in EBNF[7]. The purpose of the grammar is twofold with regard to current and potential future work:

- now: specification for the ad-hoc CSV parser;
- later: open the possibility of using an off-the-shelf parser, the grammar being one of its inputs.

The somewhat simplified grammar is shown below (many units are missing...). W3C conventions are used.

[7] Extended Backus-Naur Form

```
start     ::= requirement ( '#CR#LF' requirement )*
requirement ::= object_declaration | enum_declaration |
    feature_declaration
object_declaration ::= object_name ';;;' ( '*' superclass )?
    ';;;;'
enum_declaration ::= enum_name ';;;' 'enum' ';' enum_values
    ';;;'
feature_declaration ::= object_name ';' feature_description
superclass ::= object_name
object_name ::= identifier
identifier ::= [A-Za-z] [A-Za-z0-9_]*
feature_description ::= feature_name ';' info ';'
    feature_details ';' authority ';' time_dependency
feature_name ::= identifier
info ::= text
text ::= character*
character ::= [A-Za-z0-9,] | space
space     ::= [#x9#xA#xD#x20]
type  ::= numeric_type | nonnumeric_type
feature_details ::= 'optional'? ('sorted '? 'list of ' )? (
    numeric_type ';' unit | ( nonnumeric_type |
    object_reference ) ';' | 'enum' ';' enum_values |
    enum_reference ';')
numeric_type ::= 'int' | 'float'
nonnumeric_type ::= 'str' | 'boolean'
unit      ::= 'counter' | 'dimensionless' | 'kg' | 'm' | 's' |
    'Celsius'
enum_name ::= identifier
enum_values ::= enum_value ( ',' enum_value )*
enum_value ::= identifier
enum_reference ::= '*' ( module_name '.' )? enum_name
object_reference ::= '*' ( module_name '.' )? (type |
    object_name)
module_name ::= identifier
authority ::= priority ';' source ';' isdefinedby
priority ::= digit | 'skip' text?
source    ::= text
isdefinedby ::= text
digit     ::= [0-9]
time_dependency ::= 'Static' | 'Quasi_static' | 'Dynamic'
```

Requirements may also express abstract datatypes (such as text or numeric), associations (using '*' as a prefix), and multiplicities (using "list of"). Exact multiplicities (lower and higher bound) and concrete datatypes (e.g. those defined in JSON) are left for later stage processing. Therefore, domain experts should not expend effort on such details, while semantics often remain unattended.

5.3 Semantics

A realistic design goal for UML models is to embed semantics in the model itself. This has been consistently achieved, for instance by EULYNX DataPrep, RSM, or TRANSMODEL, by extensive use of UML notes pertaining to diagrams, classes, attributes, or associations. Notes are, as far as possible, brought to the surface of the class diagrams, and in any case remain accessible when the diagrams are published in XML or other text formats.

To enable these high standards to be achieved, there should be a strong focus on the robustness, accuracy, completeness, and specificity, of the requirements formalisation by the domain experts from the beginning of the process.

Each data requirement (object, feature, enumeration...) is identified by a name (object name, feature name...) or a short phrase that should be expressive, unambiguous, and familiar to experts, as much as possible, unique across the whole set of requirements. These names will be used as identifiers after transformation (e.g. camel-casing).

As names alone are not sufficient to define the objects, an "info" key was introduced at an early stage in the JSON schema of the X2RAIL-4 model: an "info" column was provided accordingly in the input sheets, which is relevant to each single requirement and is intended for the specific definition and full explanation of the feature. Expected values are one sentence or two, without conditionals. Since one sentence is helpful, but not a reference, we recommend to point to a public, freely accessible resource. In the RDFS framework, the annotation property rdfs:isDefinedBy is dedicated to such purposes. However, isDefinedBy may point to any resource. In our context, we need the resource to be public and published, preferrably in the shape of a URI. Moreover, annotation properties are ignored by reasoners. Consequently, a "hasPublicDefinition" key was introduced, in line with object property ontorail:hasPublicDefinition.

5.4 Authorities

The authority defining the terms of the requirement have been separated from the authority that expresses the requirement. Two columns remain associated with the requiring authority:

- a priority level, the meaning of which is somewhat ambiguous: priority of requirement in view of model extension, or priority with respect to data exchange, e.g. in case of channel saturation. In the context of OPTIMA, the second meaning applies;
- a source, that ensures traceability of the requirement and, indirectly, identifies the authority expressing the requirement.

6 Transformation and integration

Formalized requirements are intended to extend the source model (here: X2RAIL-4), leaving the existing parts unchanged. We expect extensions to be ignored by

those applications that do not require them. However, the extended model must in any case conform the original JSON schema. Processing requirements takes four steps:

informal requirements → formal requirements → pre-processing → processing → post-processing

6.1 Pre-processing

The pre-processing step of model transformation is partly automated. The automated part consists of:

- check the completeness of each single requirement, especially with respect to units or dangling references;
- suggest structures and properties matching objects and features formulated in the requirements;
- check whether features are static or dynamic, in view of expressing the time-dependency of feature values in JSON;
- express multiplicities using the JSON schema conventions.

The manual part of pre-processing then consists of:

- dealing with the warnings and errors provided by the pre-processing execution log,
- assigning requirements to existing JSON modules, and
- indicate the matching JSON structs and enums, when possible.

6.2 Processing and post-processing

Both the processing and post-processing steps are fully automated, the output of the processing is the set of extended JSON modules, and post-processing checks JSON schema conformity. The salient features of the processing are:

- offering the choice to instantiate object features 1) as attributes, or 2) as references to a reified attribute, 3) possibly reversing the dependency direction (observer pattern);
- replacing subclassing (not supported by JSON) by having the subclass referring to the superclass, rather than inheriting its attributes;
- using the observer pattern to express "dynamic" attributes (having time-dependent values). In this case, the preferred solution is to pair the Foo class with a FooState class, which holds the dynamic attributes, a single timestamp, and a reference to Foo.

7 Conclusions and further works

Shift2Rail OPTIMA project deals with the design and validation of a demonstrator of a communication platform to test new TM applications. Requirement

elicitation and transformation of the requirements into a data model is a part of the OPTIMA project. The requirement formalization and transformation toolset was developed in response to the complexity of the data model required for TMSs and the project timescales (as well as reducing the time for future work). There has already been interest from other projects and interested parties in both the formalization and toolsets (demonstrating the relevance of this work), which are intended to be shared under some sort of open source license.

By further experimenting with model extension and combination, either using the proposed, semi-automated methodology or more creative, whiteboard-based methods, we are confident that OPTIMA will achieve its particular goal, i.e. running a TMS demonstrator. The somewhat unexpected bonus is however a contribution to building the CDM, which is one of the main goals pursued by the European Commission and could be of utility to the railway industry.

References

1. Board, S.J.G.: Shift2rail joint undertaking multi-annual action plan (2015), https://shift2rail.org/wp-content/uploads/2013/07/S2R-JU-GB_Decision-N-15-2015-MAAP.pdf
2. Board, S.J.G.: Shift2rail joint undertaking multi-annual action plan (2019). https://doi.org/doi:10.2881/314331, https://shift2rail.org/wp-content/uploads/2020/09/MAAP-Part-A-and-B.pdf
3. Cecchetti, G., Ruscelli, A.L., Castoldi, P., Ulianov, C., Hyde, P., Oneto, L., Marton, P.: Communication platform concept for virtual testing of novel applications for railway traffic management systems. In: Proceedings of 24th Euro Working Group on Transportation Meeting (EWGT 2021) (September 2021)
4. FINE-2: Furthering Improvements in Integrated Mobility Management (I2M), Noise and Vibration, and Energy in Shift2Rail (Dec 2019), https://projects.shift2rail.org/s2r_ipcc_n.aspx?p=fine-2
5. LinX4Rail: System architecture and Conceptual Data Model for railway, common data dictionary and global system modelling specifications (Dec 2019), https://projects.shift2rail.org/s2r_ipx_n.aspx?p=LINX4RAIL
6. Liu, J., Ulianov, C., Hyde, P., Ruscelli, A.L., Cecchetti, G.: Novel approach for validation of innovative modules for railway traffic management systems in a virtual environment. Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit (Aug 2021). https://doi.org/10.1177/09544097211041879
7. OPTIMA: cOmmunication Platform for TraffIc ManAgement demonstrator (Dec 2019), https://projects.shift2rail.org/s2r_ip2_n.aspx?p=S2R_OPTIMA
8. Salih Dawood, O., Sahraoui, A.E.K.: From Requirements Engineering to UML using Natural Language Processing – Survey Study . European Journal of Industrial Engineering **2**(1), pp.44–50 (Jan 2017). https://doi.org/10.24018/ejers.2017.2.1.236, https://hal.laas.fr/hal-01703317
9. X2Rail-4: Advanced signalling and automation system - completion of activities for enhanced automation systems, train integrity, traffic management evolution and smart object controllers (Dec 2019), https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-4