Horizon 2020
European Union Funding
for Research & Innovation

In2Rail

In2Rail

Project Title:	INNOVATIVE INTELLIGENT RAIL
Starting date:	01/05/2015
Duration in months:	36
Call (part) identifier:	H2020-MG-2014
Grant agreement no:	635900

Deliverable D7.4

Definition of the Proof-of-concept

Due date of deliverable	30-04-2017
Actual submission date	30-04-2017
Organization name of lead contractor for this deliverable	DLR
Dissemination level	Public
Revision	Final

Authors

		Details of contribution
Authors	**Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)** Stefanie SCHÖNE	Overall author and editor General approach
	SIEMENS (SIE) Stefan WEGELE	Prototype Architecture, Generation and Discussion of Generic Use Cases Mapping of D 7.2 Requirements on Prototype Architecture
Contributors	**HaCon (HC)** Rolf GOOSSMANN Sandra KEMPF	Prototype Architecture, Generation and Discussion of Generic Use Cases
	Ansaldo Marco GIAROLI Carlo DAMBRA	Prototype Architecture, Generation and Discussion of Generic Use Cases
	Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR) Christian LINDER	Prototype Architecture
	RFI	Prioritisation of Use Cases
	TRV	Prioritisation of Use Cases

Executive Summary

The overall aim of the In2Rail project is to set the foundation for a resilient, cost-efficient, high capacity and digitalised European rail network.

There are three In2Rail Work Packages relating to Intelligent Mobility Management (I2M), one of which is WP7. This WP deals with the definition of the functional specification of future TMS / dispatching systems and validation of the results from WP8 and WP9 in respect on these requirements.

This document is the fourth deliverable in WP7 and states the way in which the proof-of-concept shall be executed. Use Cases with high priority for Infrastructure Managers were derived from the TMS Use Cases from WP7 D7.2 and a proof-of-concept prototype was sketched that, implemented in the coming time period, will be used to validate the requirements stated in WP7 D7.2. For maximising the outcome of the proof-of-concept with the limited efforts possible in the In2Rail Proof-of-concept phase, existing solutions for TMS Applications are planned to be integrated in the prototype when possible.

TABLE OF CONTENTS

Abbreviations and Acronyms

Abbreviation / Acronyms	Description
AF	Application Framework
API	Application programming interface
ATO	Automatic Train Operation
BDA	Big Data Architecture
BLOB	Binary Large Object
Canonical Model	Hierarchically structured object data model; developed and specified in WP8
COTS	Commercial off-the-shelf
DBMS	DataBase Management Systems
(H)DFS	(Hadoop) Distributed File System
D x.y	In2Rail Deliverable x.y from Work Package x
DGE	Data grid explorer
EDW	Enterprise Data Warehouse
ETA	Estimated time of arrival
ETL	Extraction, Transformation and Loading of Data
HAFAS	HaCon Fahrplan-Auskunfts-System; Timetable and information management system developed by HaCon
IL	Integration Layer
IM	Infrastructure Manager
I²M *	**I**ntelligent **M**obility **M**anagement: information developed as a strategically critical asset: • A standardised approach to information management and dispatching systems enabling an integrated Traffic Management System (TMS). • An Information and Communication Technology (ICT) environment supporting all transport operational systems with standardised interfaces and with a plug and play framework for TMS applications. • An advanced asset information system with the ability to 'nowcast' and forecast network asset statuses with the associated uncertainties from heterogeneous data sources.
IMDG	In-Memory Data Grid
JSON	JavaScript Object Notation
LTG	Live Train Graph
POSIX	Portable Operating System Interface; family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems
RBC	Radio Block Centre, introduced by ETCS specifications
RU	Railway Undertaking
SW	Software
TIR	Temporary infrastructure restrictions
TMS	Traffic Management System: a traffic control-command and supervision/management system, such as ERTMS in the railway sector.

Abbreviation / Acronyms	Description
TPS	Train Planning System (system developed by HaCon)
TRL	Technical Readiness Level
UIC406 leaflet	The UIC Leaflet 406 provides guidelines for calculating capacity. The approach is to calculate capacity consumption by compressing a timetable and to evaluate the number of possible train paths for a line, node or corridor.
WP7	Work Package 7: System Engineering of Intelligent Mobility Management (I^2M) of In2Rail.
WP8	Work Package 8: Integration Layer of Intelligent Mobility Management (I^2M) of In2Rail.
YARN	Yet Another Resource Negotiator

** Definition extract from §Common Glossary of the* [IN2RAIL D7.1] *deliverable of In2Rail.*

1 Background and Objective

This document constitutes the first issue of Deliverable D7.4 "Definition of the Proof-of-concept" in the framework of the project entitled "Innovative Intelligent Rail" (Project Acronym: In2Rail; Grant Agreement No 635900).

The overall objective of Work Package 7 – WP7 – is to provide the specification to validate the Intelligent Mobility Management (I²M) open integrated platform for Traffic Management Systems (TMS) and dispatching systems of the future and validate the outcome from WP8 and WP9. WP7 covers three topics, which come at different development stages of the future Traffic Management System:

- Task 7.1: to carry out the requirement analysis;
- Task 7.2: to specify a Standard Operators' Workstation allowing the display and control of all services and functions applied in an integrated traffic control centre;
- Task 7.3: to validate an integrated I²M Technical Readiness Level (TRL) 3 proof-of-concept built around the Integration and Application Layers, the Demand Management functionalities and the nowcasting and forecasting of the network assets status.

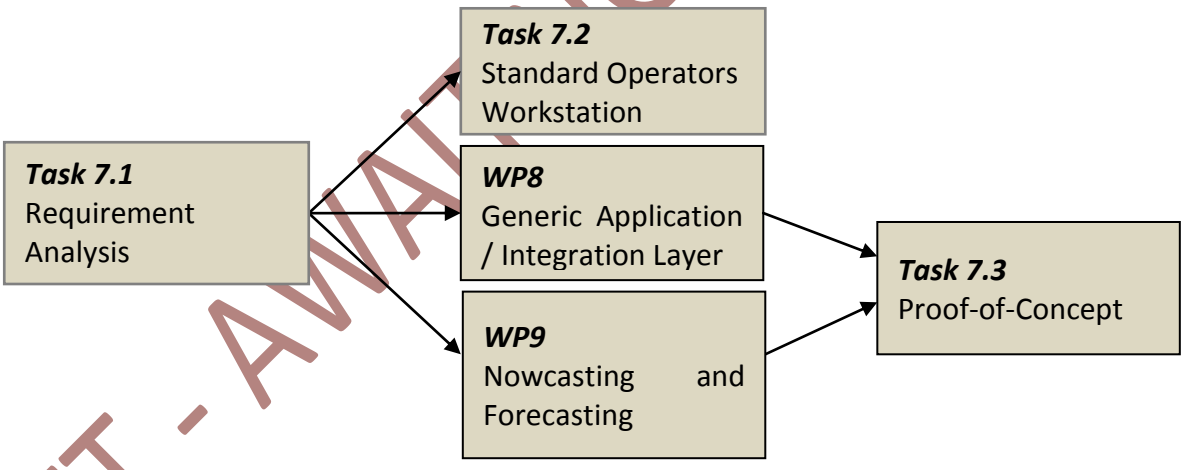

Figure 1.1: Simplified view on integration of WP7 with WP8 and WP9

The objective of WP7.3 is, as Figure 1.1: shows, to provide a reliable proof, that the selected architecture for Integration Layer and Application Framework in WP8 and the asset status functionalities developed in WP9 are sufficient to fulfil requirements of a future Traffic management system stated in Task 7.1.

Use Cases shall be defined that will be demonstrated in the proof-of-concept at the end of the project In2Rail, when in the last phase of Task 7.3 the prototype shall be validated against the system requirements from D7.2, as shown in Figure 1.2.

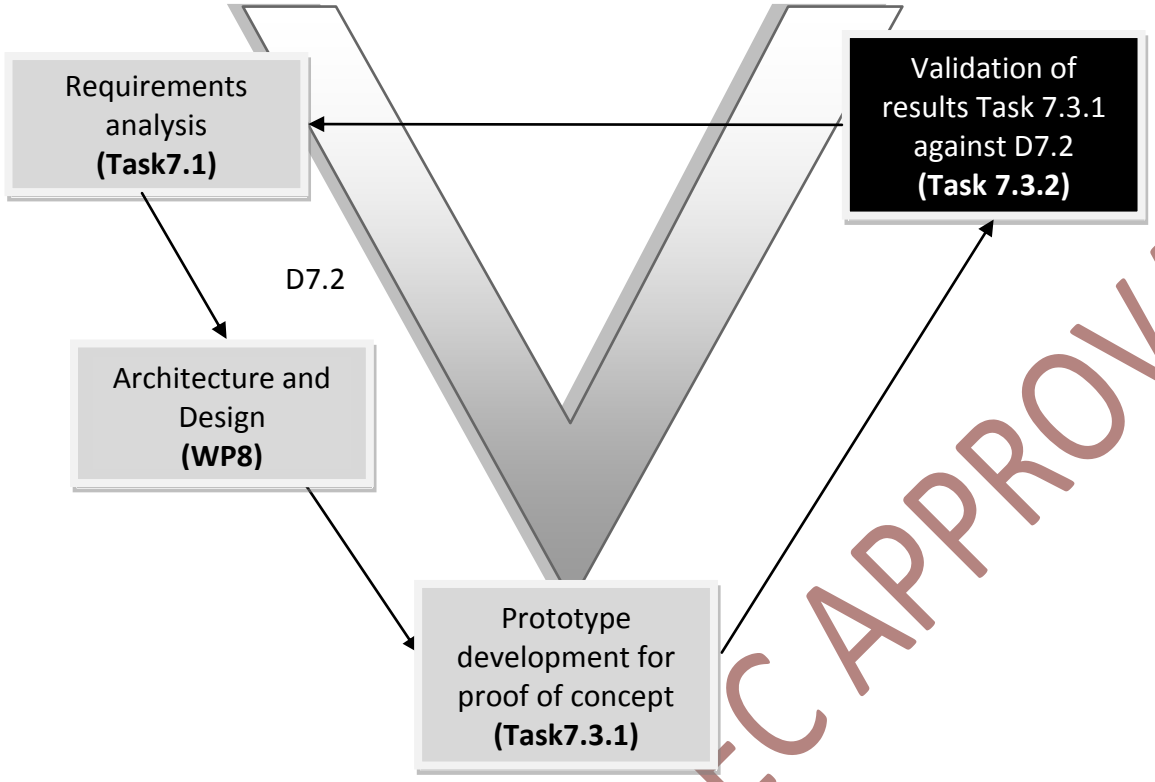

Figure 1.2: Development process as suggested in D 7.2 [IN2RAIL D7.2]

Deliverable D7.4 is the result of the first step of Task 7.3, defining the proof-of-concept method as the use of a demonstrator. Therefore Generic Use Cases were derived from the previous Deliverables of WP7 and an according prototype architecture is sketched that is needed to validate the requirements from Task 7.1. As the work, both on the proof-of-concept itself, and in the presupposed WP8 and 9 are ongoing, this deliverable represents a state of work of all of these three Work Packages. Thus alterations of the prototype architecture and the Use Cases cannot be excluded especially as to date there has not been any official Deliverables of WP8.

This document is the result of several workshops with the participation of all Task 7.3 members (SIE, DLR, HC, ASTS, TRV, NR), starting in June 2016. There and with the aid of consultations of Infrastructure Managers, the most interesting Use Cases for the IMs were derived and the most appropriate software architecture was developed, which allows validating the system against as many of the requirements as affordable with the limited amount of resources and time available in this work package.

2 General Approach on In2Rail Proof-of-concept

In previous tasks of WP7 over 1000 requirements have been collected on the future Traffic Management System (TMS) (see [IN2RAIL D7.1], [IN2RAIL D7.2]. The Integration Layer (IL) and Application Framework (AF) themselves do not provide any user specific functionality, but enable the software producers to develop and integrate their software functions into a big scale TMS.

From a high level perspective, the requirements on the IL can be structured and are planned to be proven in the following way:

- Achievable performance for TMS specific use cases as considered in the next chapters. To evaluate them, real applications with real data for a middle European country scale shall be used for the evaluation of response times with high load;
- Extensibility of the IL – can be tested by:
  - Integration of several components by different providers,
  - Covering several aspects of the data model (topology, timetable, restriction management, etc.),
  - Data model extension and check for backwards and forward compatibility (old client - new service provider, new client – old service provider);
- Supported platforms, estimated long term availability – by market analysis and the IL-complexity;
- Completeness of the canonical model – requires practical evaluation basing on several Infrastructure/Topology data sources. It will be checked only for one data set;
- Failover, reliability – shall be analysed for developed prototype by manual check for single point of failure,
- Security – will be analysed manually by check of the product specifications and IL-approach.

On the one hand the more functions and products will be integrated into the prototype the more reliable the evaluation results for the requirements are. On the other hand in this work package the time horizon and the available resources are limited. To design the most appropriate prototype architecture the following approach was used (see **Figure 2.1**).

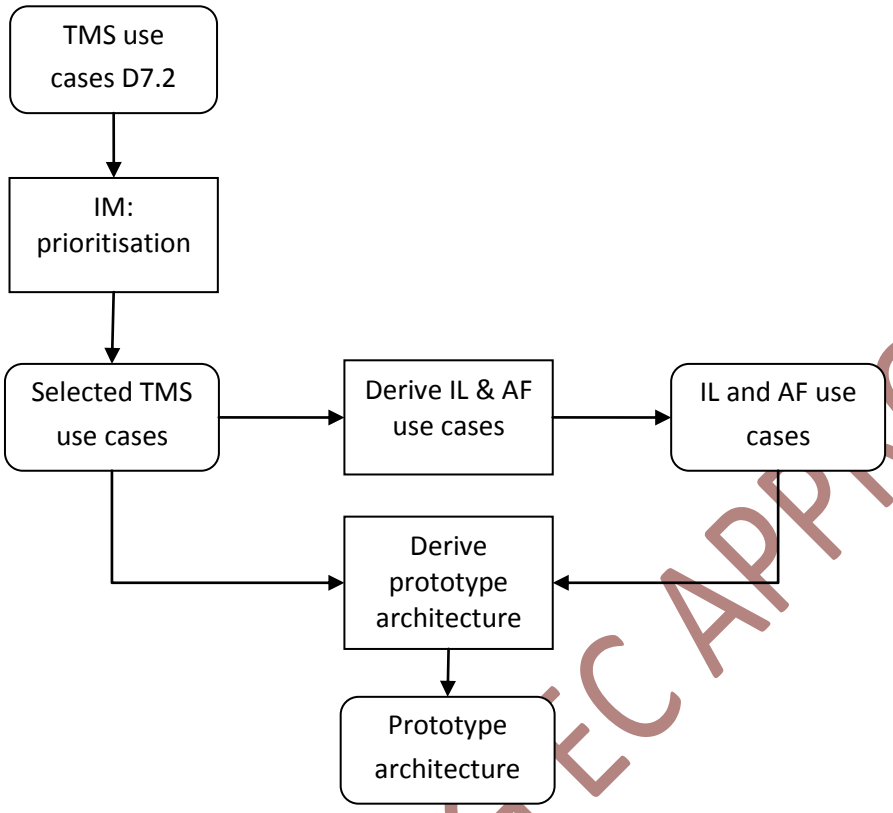

Figure 2.1: General approach for proof of concept definition

Starting with the TMS specific use case definition provided in [IN2RAIL D7.2] we asked the Infrastructure Managers participating in In2Rail for prioritisation of these use cases in order to reduce the scope of the prototype to the most important functions in TMS to validate in the proof-of-concept. The prioritisation supplied by RFI and TRV are shown in Table 2.1.

ID	Use Case	RFI	TRV
UC 1	Manage Maintenance information	**7,6**	
UC 1.1	Generate Day of Operation Maintenance Plan	7	
UC 1.2	Request to modify the Day of Operation Maintenance Plan	7	
UC 1.3	Monitor and tracking maintenance work	7	**1**
UC 1.4	Request for extension of time window for a work	7	
UC 1.5	Propose optimal working window for a work	10	**0**
UC 2	Manage Very Short Time Request	**8,3**	
UC 2.1	Manage request to suppress a service	8	
UC 2.2	Manage request to cancel a service	8	
UC 2.3	Manage request to modify running time of a planning service	8	
UC 2.4	Manage request to modify route of a planning service	8	**4**
UC 2.5	Manage request to link two planning service	10	
UC 2.6	Manage request to add a new service	8	**3**
UC 3	Manage Real time traffic plan	**9,6**	
UC 3.1	Monitoring and audit of the train movement	10	**2**

ID	Use Case	RFI	TRV
UC 3.2	Add real rolling stock and crew to Real time traffic plan	9	
UC 3.3	Train movement forecasting calculation	10	**8**
UC 3.4	Conflict detection	10	**5**
UC 3.5	Conflict resolution	10	**9**
UC 3.6	Modify the behaviour of the trains by minor disturbance	9	
UC 3.7	Modify the behaviour of the trains by major disturbance	9	**6**
UC 3.8	Automatic route settings	10	
UC 4	Manage & Monitor Train Traffic & infrastructure	**10,0**	
UC 4.1	Receive information of Train Traffic & infrastructure status	10	**10**
UC 4.2	Send operation commands to IM control systems	10	
UC 5	Manage train traffic information distribution	**9,0**	
UC 5.1	Send information to the stakeholders	9	**7**
UC 6	Analysis and tracing of information	**7,0**	
UC 6.1	Analysis of registered information about running of the trains	7	
UC 7	Manage temporary traffic restrictions	**7,5**	
UC 7.1	Add or modify a temporary restriction to the real time traffic plan	7	
UC 7.2	Remove a temporary restriction from the real time traffic plan	8	
UC 9	Use Cases provided by WP9 - yet to come		

Table 2.1: TMS Use Cases from [IN2RAIL D7.2], prioritised by RFI and TRV, being 10 the most and 0 the least important

Although the approach of the prioritisation was handled different by the two IMs (TRV picked only the ten most interesting whereas RFI gave a priority to each use case), it shows that they regard the same Use Cases as the most important to show in the proof-of-concept. As follows from the table the Infrastructure manager's highest priority belongs to monitoring and management of the traffic and infrastructure. In addition the IMs are interested in supporting functions (e.g. conflict detection and resolution) as future intelligent functionality.

On the basis of this prioritised Use Cases there was a workshop with the Task 7.3 participants in which so called "generic Use Cases" were derived from the (according to Table 2.1) most important D7.2 Use Cases. These "Generic Use Cases" represent the IL & AF use cases according to Figure 2.1.

The developed generic use cases that were selected for the future development of a prototype are shown in Table 2.2.

ID	Description
1	**Real-time status updates** coming from field (signalling/train control and train position updates)
2	- Massive data collection from field objects for feeding asset forecasting models; - **Switch degradation** forecast is pushed to TMS via Interface Plugin; - TMS application to create temporary infrastructure restriction (non safety relevant) as part of the production plan, according to business rules; - TMS staff decides that a TSR is created; - train running forecast calculation performed accordingly; - conflict detected **(minor disturbance)** and conflict resolution provides solution options (sand-box) for decision support; - one solution option selected by dispatcher, production plan being updated and disseminated to staff/systems/stakeholders
3	The same as Generic Use Case 2 with **major disturbance**, i.e., Train Operators to be involved into decision making (dynamic demand interface)
4	TAF/TSI ad hoc **freight request between IM** (JBV and TV or between NR and SNCF Réseau): Inbound request, Path creation, Path negotiation and dissemination (both neighbouring IMs involved)
5	**Massive data sent by components via IL/AF** (e.g., re-sync of TMS after shutdown period and take-over by redundant system; or initial national traffic status after re-start of PIS components)
6	**Massive data inbound**: Update of initial daily production plan from planning system
7	**Estimated Times of Arrival (ETA) are sent to ATO trackside** after update of production plan.
8	- External consumer application request updates prognosis (e.g. ETA of a train), - a service plug-in delivers the data request to the information manager, - an application is launched to perform prognosis - by that calling asset status forecast as pre-condition - resulting ETA is then handed back to the information manager, and finally a service plug-in communicates back the result to the originating consumer application through the integration layer (**Example from** Description of Work in **Grant Agreement**)

Table 2.2: Generic Use Cases

By executing these generic use cases with a prototype that fulfils the requirements from [In2Rail D7.2], the most important functions of the TMS shall be shown.

Again in a workshop with all Task 7.3 participants a prototype architecture was designed that shall provide the functionalities to show all generic use cases. For the description of the planned prototype architecture see chapter 3.

Several aspects have to be left out for a start, granting the core functionality, but could be added when there is enough time and resources left towards the end of the project (the numbers correlate with the generic use cases in Table 2.2):

- 1 and 3: the main focus lies on the nowcast aspect instead of forecast;
- 2: will be left out, as the functionalities should be part of 3;
- 4: implementation without standardised TAP/TAF-interface;
- 5: will be managed as a restart of all clients;

- 6: could be covered with an import of a complete data set of a Middle European country or with providing a client with massive data, ensuring high frequency and low latency;
- 7: there will be no ATO system simulated; the data shall be sent to a black box;
- 8: a (simulated) passenger information system shall be integrated.

As the approach of the proof-of-concept is to show functional use cases, and in the respect of the limited time and effort that is available in Task 7.3, several aspects cannot be tested separately in Task 7.3 (if they are not provided by the used soft- and middleware solutions anyway):

- Security (possible only by manual inspection);
- Authentication and Authorization (possible only at application level) TMS-appropriate scaling (not every use case will be demonstrated in the scale of a whole middle European country);
- Human Factors issues (Usability, Customer Comfort).

3 Planned Prototype Architecture

In a workshop the Task 7.3 participants developed a functional system architecture with all modules that are needed to show the generic use cases (see Chapter 2) and therefore proofing if the requirements set in former WP7 works can be fulfilled. The functional view of the planned prototype architecture is shown in Figure 3.1.

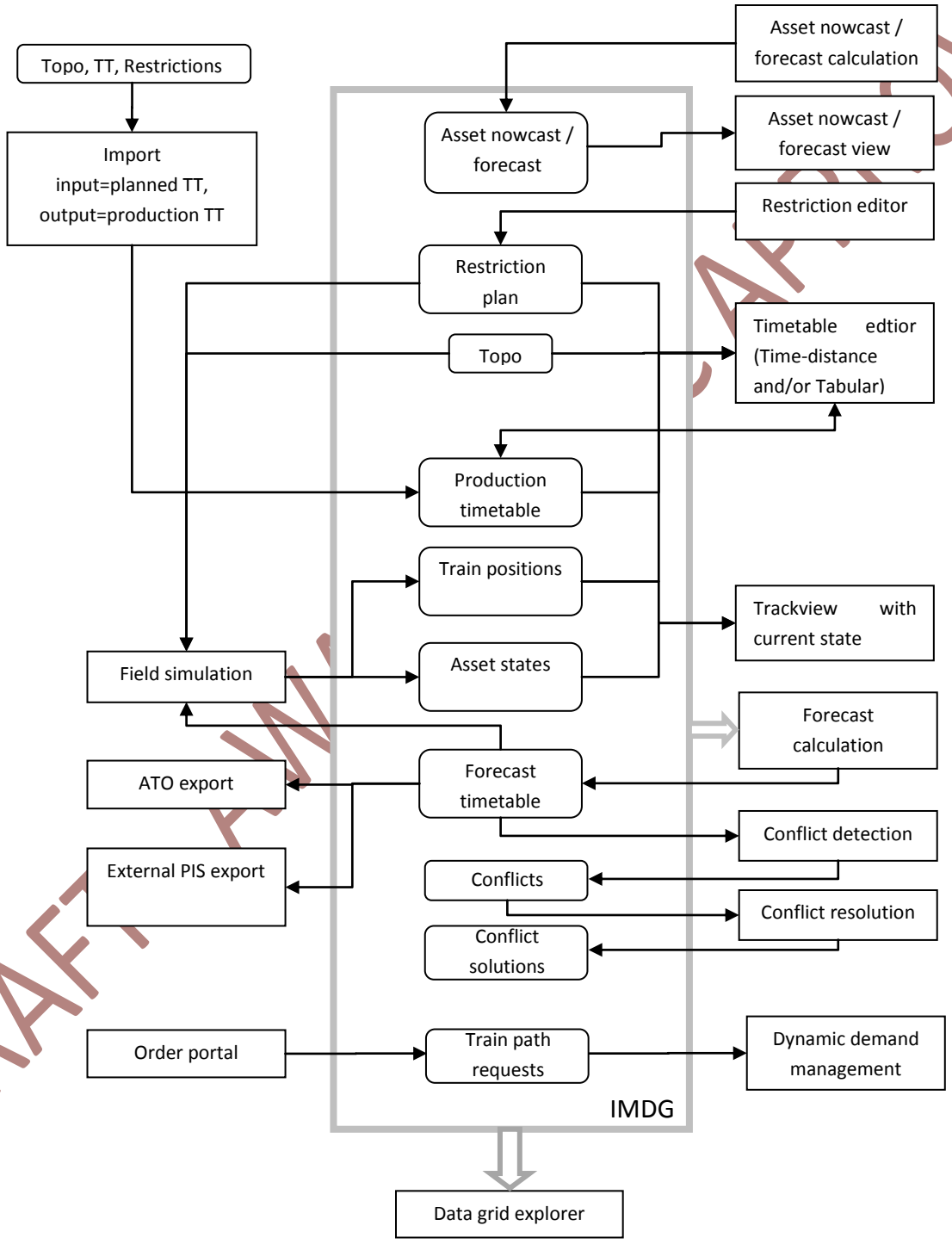

Figure 3.1: Planned prototype architecture

The planned prototype architecture covers wide areas of the canonical model, enables validation of performance requirements and evaluation of flexibility due to integration of existing productive software from several companies. In general it splits up into four main parts:

- The central data storage and distribution module based on **In-Memory Data Grid** (IMDG – grey box) with datasets in it like “Forecast timetable”, “Asset nowcast” or “Train positions”.
- The surrounding **functional modules and tools** to work with the data as e.g. “Importer/Exporter tools”, “Asset nowcast calculations” or “Timetable editors”.
- A **data grid explorer** tool to browse the data inside the IMDG. This tool will be capable to verify that all datasets inside the IMDG are consistent in format even if they originate from heterogeneous tools and vendors.
- A **data grid wrapper** (not in picture, but implied by the arrow signs) to make sure, that all data streams inbound/outbound the data grid are implemented in a standardized approach, that the data formats are consistent and to ensure multiple other features like proper exception handling or data transactions cleanly.

The general approach to provide evidence that this specific architecture will stand the proof of concept is to implement the main parts, validate them by testing the use cases (D7.2) and measure their performance with sample or real data. This will demonstrate that the chosen architecture and the chosen technologies are a suitable combination of real modules to form the abstract “Integration layer” as described in WP8.

The following chapters will give a more detailed insight about the status and approach of the implementation of software from these four main IL parts.

Again, it has to be taken into account that the works base on the work status of other work packages, namely Deliverables 8.3, 8.4, 8.6 and 8.7, which will be submitted in a final form after this document.

3.1 Data storage and distribution Layer (In-Memory Data Grid (IMDG))

As shown in Figure 3.1, the IMDG element is the central module to store all datasets in. As an example: train positions, timetable data or predictions are stored inside this unit, while these datasets are produced outside the IMDG within their specific environments e.g. timetable editors or simulation frameworks. Due to the variety of surrounding software modules and their different implementations, it is necessary to provide a common gateway to store data inside the data grid and distribute to all subscribers. Some SW components might be written in Java, others in C++, some might store their data in XML and other formats. To avoid a mixture of data formats and unstructured and unusable data within the system the WP8 introduced a generic wrapper to encapsulate the data grid. This wrapper will offer an open programming interface (API) to all applications and ensures that the data

inside the IMDG is consistent. The wrapper as an integral part of the "Integration Layer" will be discussed in the following chapter in more detail.

As a choice from WP8, the central data storage system in In²Rail will be the IMDG-like platform. This choice was based on several positive characteristics the data grid technology provides. One main aspect here is that IMDG provides seamless reliability, high performance for requests and a build-in change distribution. IMDGs are therefore one of the best hands-on choices for real time applications, as In²Rail aims to go for.

Another aspect of data grids is their ability to be used in distributed environments and network clusters. IMDG nodes automatically search for neighbouring nodes and join up to clusters. That is where the term "grid" originates from. Such a cluster or grid makes use of several fundamental concepts in computer science like load balancing, data redundancy, distributed data objects, fail safety (data in dying nodes is redistributed), etc. Figure 3.2 illustrates these aspects.

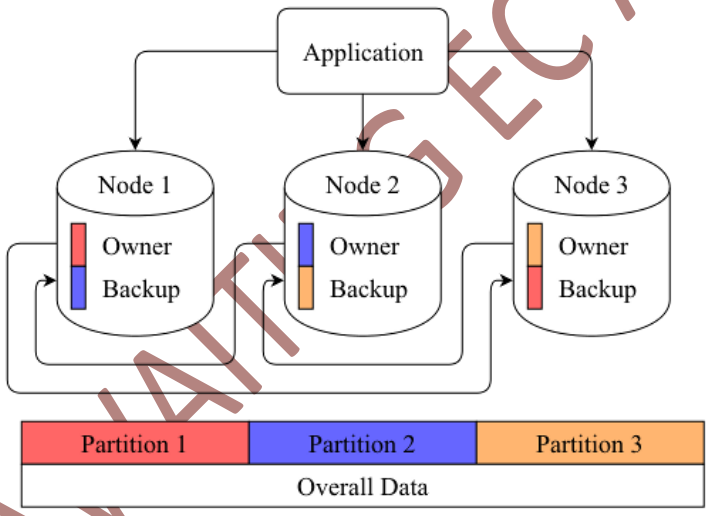

Figure 3.2: Distributed IMDG with 3 nodes showing data redundancy in multiple partitions

As part of the proof of concept, Task 7.3 shall provide evidence if existing IMDG implementations are a suitable part of the "Integration Layer" as proposed by WP8.

3.2 Data Grid Wrapper (Middleware)

As described in Figure 3.1 one core element is the IMDG – the In-Memory data grid. As shown in this figure, multiple data types (such as asset nowcast, topology data and/or production timetables) will be stored inside the data grid and shared with the surrounding modules like "field simulation", "timetable editor" and/or "restriction editors" to just name some of the applications.

To provide a proper way to manipulate and share the data inside the IMDG amongst all applications it is necessary to encapsulate the specific IMDG implementation from the

outside world. Therefore a data grid wrapper was introduced to receive the commands from the application and pass them through to the data grid as shown in Figure 3.3.

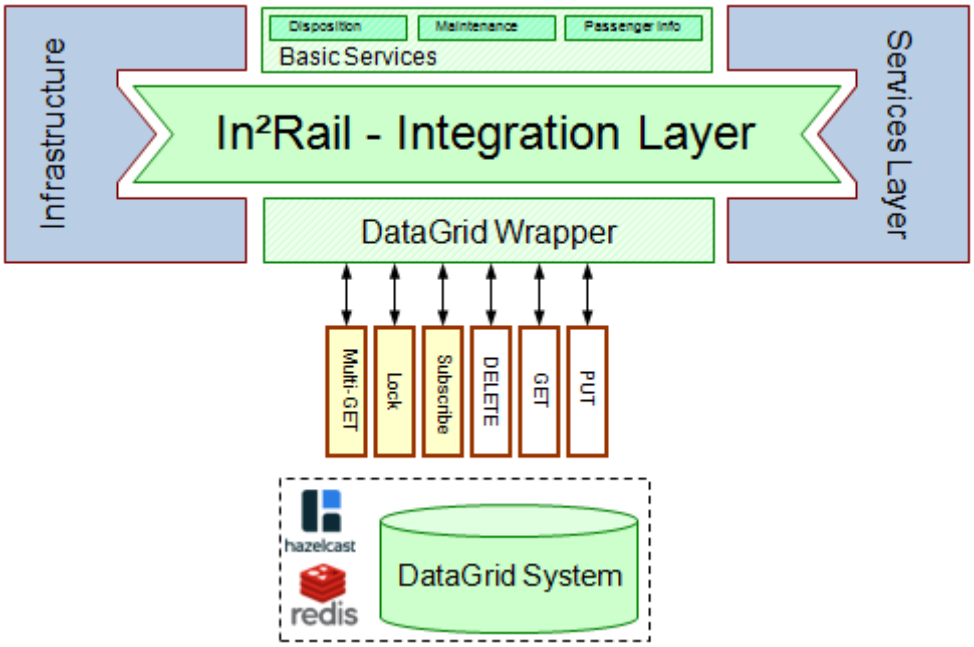

Figure 3.3 Introduction of the Data Grid Wrapper

The data grid wrapper is allocated between the applications in and around the integration layer and offers commonly known data grid services to the users, such as PUT, GET, DELETE and more. This enables a standardised usage of the data grid from all applications, ignoring which specific data grid system is running on the server. In this project it is intended to proof, that two different technologies (HazelCast and Redis) provide the data grid service without changing the application code, since they are using the wrappers API.

Figure 3.4 shows a more detailed view of the wrapper. In the top section it separates the offered commands (Wrapper-API) in three groups:

- The first group provides basic data manipulation functionalities to PUT, GET (retrieve and search), DELETE and Multi-GET data from the grid;
- The second set of functions is used internally by the wrapper itself to process the incoming commands. E.g. if a user tries to get an object with object address: *"/tms/timetables/timetable1/*"* the internal regular expressions engine (Regex) is used to process the address string. Another feature is responsible to compare the given address strings with the canonical data model and its mapping inside the data grid. This feature is necessary since data grids operate with key-value formatted data formats. Furthermore, the wrapper provides logging functionality to log data manipulation;
- The third group of wrapper API functions are related to object handling of the data objects. With these functions it will be possible to lock specific objects and prevent editing of their values to enable transactions within the application ecosystem. It can

be used to safely edit data objects without the danger that other applications change the values at the same time. Furthermore, it will be possible to subscribe to object change events as well as a system to manage permissions. It is also possible to retrieve Exceptions (error messages) from the data grid in case they occur. The wrapper has the functionality to safely pass these exceptions from the data grid system to the causing application.

In addition to the functional API offered by the wrapper, the wrapper will be responsible to manage the mapping of the canonical model of the object data to the key-value structured data format inside each and every data grid. This is represented by the blue drawings on the right hand side, where hierarchically structured object data is mapped into <k,v>-pairs.

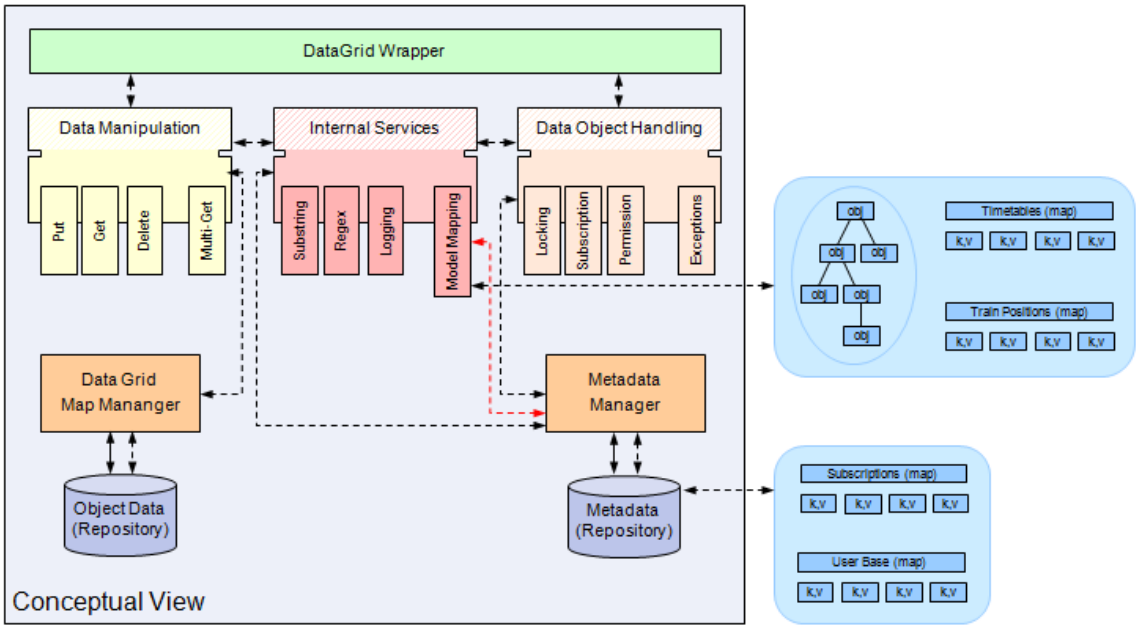

Figure 3.4: Detailed view of the data grid wrapper

As already mentioned in the previous chapters, the data outside the data grid will be structured in accordance to a canonical data model. Since, inside the data grid the data has to be formatted in key-value pairs, an appropriate mapping functionality has to be applied. This mapping is organized by one of the wrappers internal services. For this reason the wrapper itself will create data objects within the data grid and store it in a meta data repository - different from the data objects (application objects) repository. This meta data repository will also store information to support wrapper functionalities such as subscriptions, user management and other meta data. Therefore, the functionality of managing meta data as well as object data is necessarily allocated at the wrapper as well. Two modules, the “MetadataManager” and the “ObjectDataManager” will do the trick.

Finally Figure 3.5 demonstrates an additional technical view. Here, it is shown which concrete classes are involved in the wrappers operation. The main Wrapper-API is offered to the outside applications via an object with public methods (.put(), get(), delete(), getAll(),

lock(), subscribe(), etc.). An additional EventListener object can be retrieved by the application by the subscribe method. This object fires events inside the applications whenever subscribed data objects are added, changed or deleted. In case a data grid exception occurs (data grid failure) this failure is passed through to the application to let them know about this issue to enable them to react properly to it. E.g. Exception: "Object address is badly formatted", Exception: "Data Grid Server does not respond" and others. This ensures, that every exception within the data grid or the wrapper is correctly communicated to the applications.

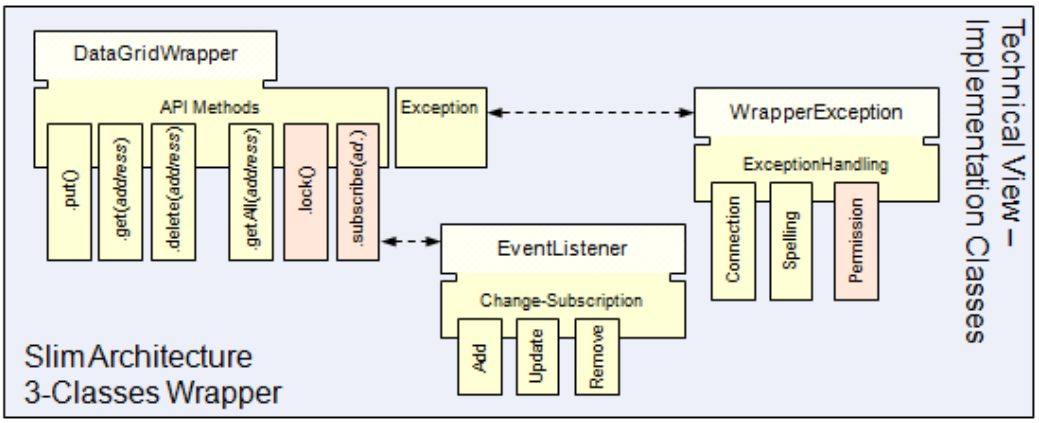

Figure 3.5: Technical view of the data grid wrapper

3.3 Data grid explorer

The data grid explorer (DGE) shall cover several aspects:

- Proof, that the selected architecture is usable on different operation systems (at least Windows, Linux);
- Proof, that it runs on different hardware (x86, ARM), which is not naturally for binary protocols;
- Allow monitoring of any data integrated into IMDG;
- Allow modifications of any data from IMDG;
- Allow validation of the data content according to the schema defined in the Canonical model;
- Proof completeness of the IMDG-Wrapper;
- Proof, that the selected architecture can use different IMDG-implementations (e.g. Hazelcast, Infinispan, Redis) without modification of the client code;
- Validate transactional functions.

The software itself is similar to Windows explorer or numerous similar programs for Linux (Nautilus, Nemo, Thunar etc.).

It represents the hierarchical canonical model as a tree and allows representation of key-value-pairs as readable and editable JSON (JavaScript Object Notation) text (see Figure 3.6).

This tool is very important during the next phases of Shift2Rail, as it provides a valuable debugging technique for integration of applications into the Integration Layer.

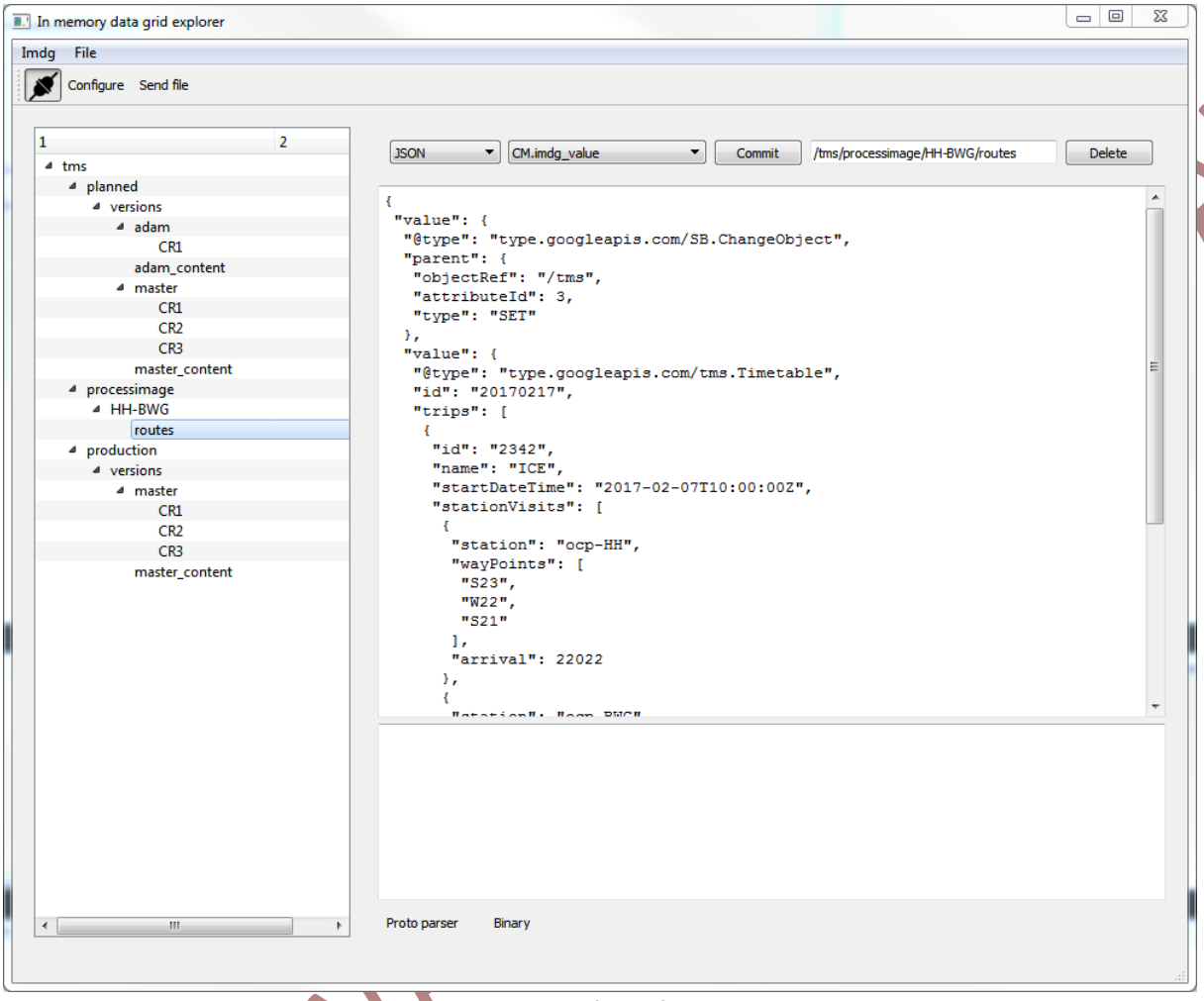

Figure 3.6: Possible user interface of the Data Grid Explorer.

3.4 Restriction editor

Various types of temporary infrastructure restrictions (TIR) can be managed using the restriction editor module which consists of a service based business logic and a specialised Client component with appropriate graphical user interface. Typical TIR types are track/line blockage, temporary speed restrictions, shunting areas, non-availability of power etc.

The restriction editor allows for configurable filtering and navigation through the set of TIRs currently available in the system as well as entering and managing detailed TIR information with respect to time, location and related types of impact.

A set of configurable business rules apply certain validation checks in order to create related messages in the case of any inconsistent or missing information. Once validated successfully by the module, TIRs are made available through the Integration Layer for other subscribing components.

TIRs can also be seen in configurable representations on the Live Train Graph, and the Track View.

3.5 Timetable editor

The timetable editor module is responsible for serving with business logic for planning/entering train schedules and applying related validation rules. A central Client component with graphical user interface in tabular form is attached to this module. Other graphical Client components such as e.g., Live Train Graph (see next section) are making use of the editor module at the same time in order to allow for "graphical editing".

The train schedules with their time and location references can be planned/entered for various types of trains together with associated rolling stock characteristics. Typical train types are passenger, freight, technical trains etc. In Traffic Management Systems, it is required to manage train schedules in different layers to e.g., allow for viewing and using the published timetable while applying dispatching decisions to the dispatcher's timetable at the same time. In addition, timetable data needs to be available in groups of train schedules for supporting the traffic management process with schedule libraries for various needs. The timetable editor includes the management of schedules in layers and groups.

The timetable editor also allows for configurable filtering and navigation through the set of train schedules ("timetable") currently available in the system as well as entering and managing detailed schedule information with respect to time, location, route and other associated operational information.

A set of configurable business rules apply certain validation checks in order to create related messages in the case of any inconsistent or missing information. Once validated successfully by the module, train schedules are made available through the Integration Layer for other subscribing components.

3.6 Live Train Graph (LTG)

One classical graphical representation of train schedules is showing train diagrams in a two dimensional time-distance diagram. The geography axis is representing the line section on which the scheduled trains will be running. The LTG allows for configuration of the geography axis being shown horizontally or vertically which addresses the different needs of different national railway systems in Europe. Besides the train schedules and a moving line representing the actual time, the LTG also visualizes TIR information as well as various types of conflicts in relation to trains schedules and/or TIR.

The LTG contains several functions, e.g.:

- Seamlessly zooming and panning, configurable orientation;
- Flexible layout definition;

- Selecting visible objects for editing or further information retrieval;
- Filtering;
- Changing objects graphically.

An assistant functionality is available on the LTG as well. If the assistant is switched on, the user can see additional information when he moves with the mouse over the LTG view. Within the LTG the dispatcher can use additional functions to change the train path, e.g., in case of disruptions. Besides others, this comprises:

- Moving of a complete train schedule or a part of it in time;
- Changing tracks/routes;
- Changing single arrival/departure or stopping times at a station;
- Changing stop types;
- Cancelling stops or parts of the train schedule.

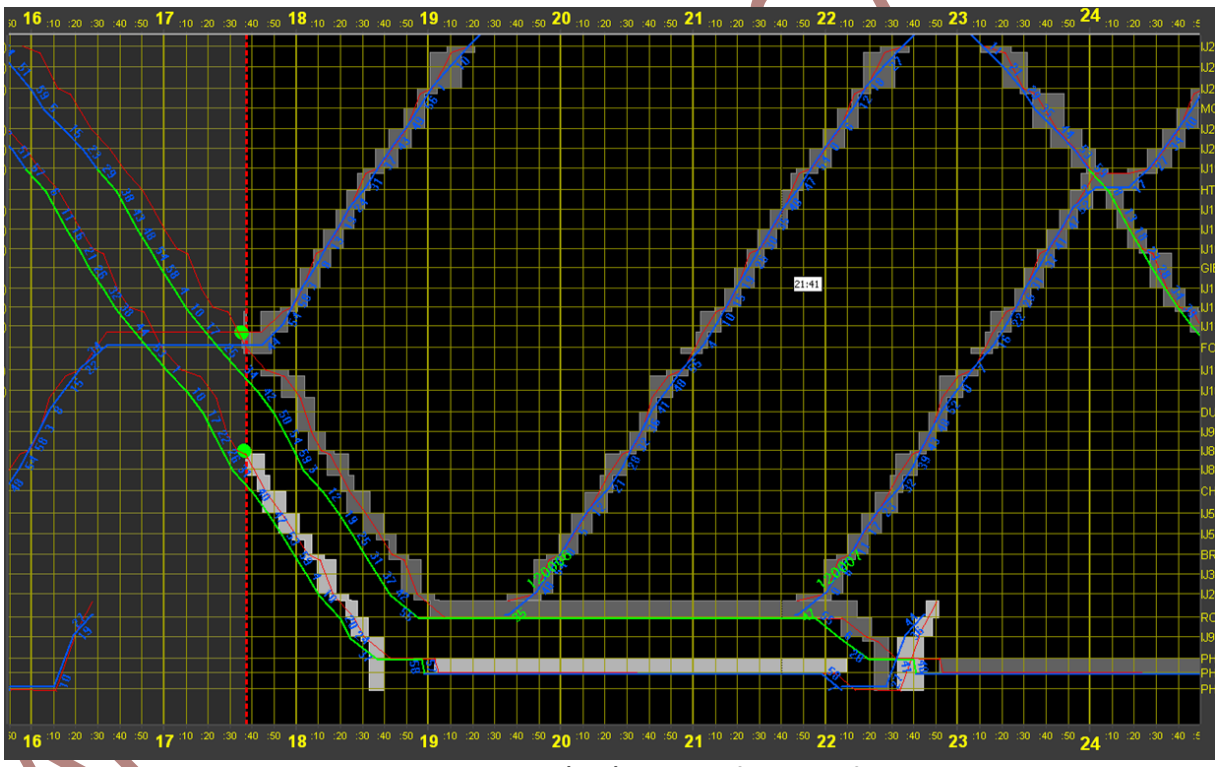

Figure 3.7: Live Train Graph (LTG) including forecast information

The red train lines in the LTG (see Figure 3.7) are representing the train running forecast and the blue and green ones are referring to the dispatcher/target timetable.

3.7 Track View

Within the Geography Editor the users can choose between a network or a line display and even view the tracks along the routes and in the train stations in microscopic detail. Line closures, construction sites etc. can be seen at a glance (see Figure 3.8).

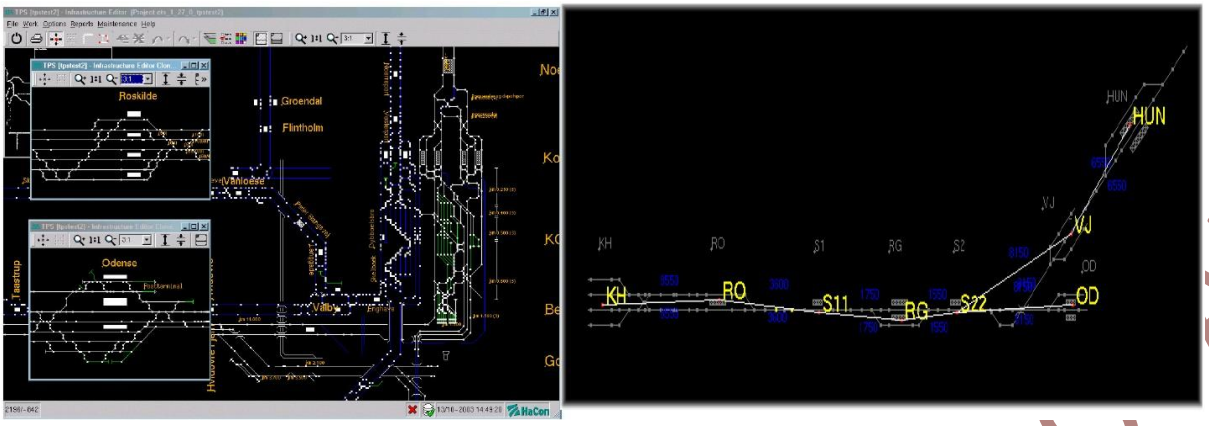

Figure 3.8: Track View in the Geography Editor

The infrastructure data model comprises, besides others:

- Nodes and Edges including geometrical attributes as e.g., lengths, curvature, gradients as well as air resistance factors (e.g. for tunnel sections);
- Switches;
- Signals and Signal Types with parameters for mapping the different signaling system behavior;
- Operational Points, Stations and Platform Tracks / Sidings including classification and typification;
- Blocks, Route sections and other types of sections including speed values for different train profiles and line characteristics as, e.g., power supply types, max. load, gauge limitations, operating conditions;
- Areas of various types for mapping of e.g., interlocking areas or area definitions for commercial or organizational purposes;
- Additional drawing elements.

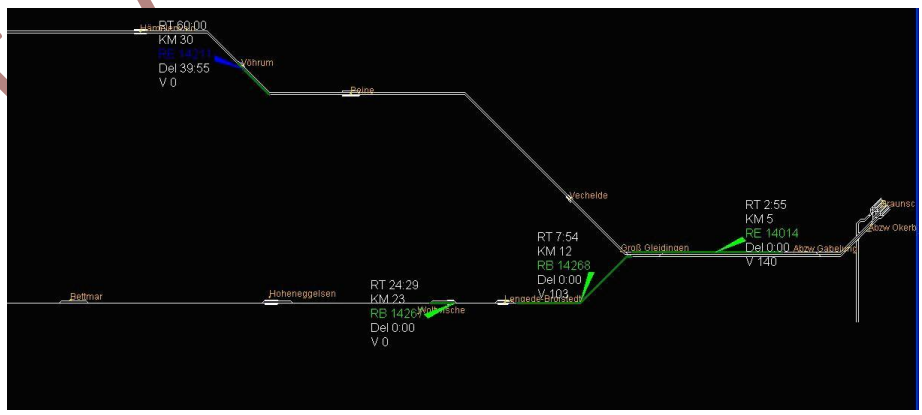

Figure 3.9: Detailed Real-time infrastructure view

Figure 3.9 shows the real-time infrastructure view in detail.

3.8 Task management (for dynamic demand)

In order to enable a controlled traffic management process, the use of a workflow module is required. Depending on the complexity of a task in terms of numbers of involved train schedules or parties to communicate with, different technical means are available in relation to workflow support:

- communication items as part of the object model;
- object status engines with configurable status transition rules;
- generic tasks with one or more assigned objects;
- worksheet based tasks with one or more assigned objects;

The tasks comprise information for identifying and describing it as well as a status engine with configurable status transition rules including pre-conditions and actions triggered by certain status transitions. The idea is, e.g., for disturbance scenarios, to let one or more dispatchers prepare different solution scenarios within a "sandbox" in order to have a final assessment of the outcomes for decision making. These sandboxes are attached to tasks describing the task and providing workflow status information and responsibilities as well as other operational or business information. At any time, data updates occurring for the original capacity objects can be merged into the prepared solution scenarios up to the time when one of the scenarios is applied to the live production plan. Making a scenario effective is reflected by a status transition of the related task.

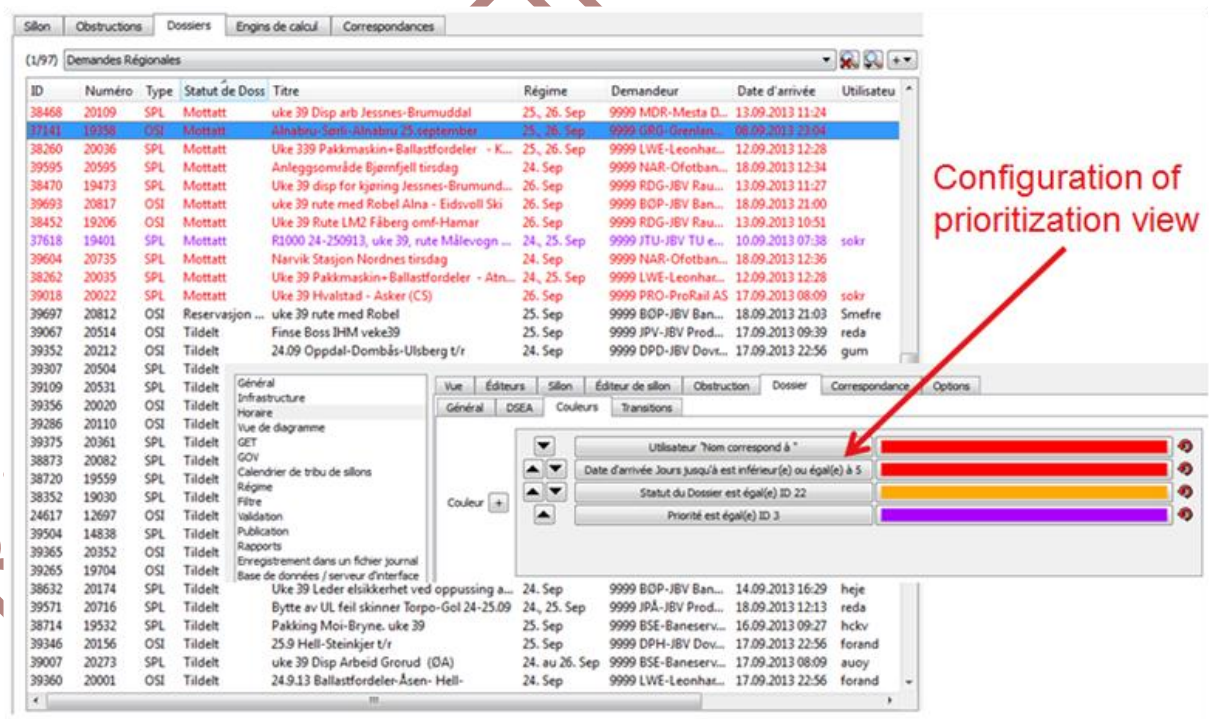

Figure 3.10: GUI of client module for management of Tasks/Worksheets

According to the different detailed processes to be supported within traffic management, different types of workflows including the related access rights, rules and actions to be

triggered can be configured. One example of a possible implementation is shown in Figure 3.10.

3.9 Forecast calculation

The forecast calculation module comprises a precise dynamic runtime calculation engine. Forecast calculations are performed automatically, e.g. after processing of train position updates or modifications of a dispatcher's schedule. The calculation itself makes use of a route search and time calculation algorithm being "guided" by the current schedule of a train and influenced by associated constraints. These constraints can be related to assigned

- topology or infrastructure elements such as planning locations, tracks or sidings;
- schedule elements such as given arrival/departure times or commercial/technical dwell times;
- rolling stock as e.g., changed load or number of traction units;
- different types of Temporary Infrastructure Restrictions (TIR);
- other timing rules.

The calculation algorithm is taking into account rolling stock driving dynamics and infrastructure related parameters such as maximum speed, various speed profiles, gradients and resistances as well as different electric power modes (see Figure 3.11).

From the calculated times, the track or block occupation times are derived for capacity consumption measurement and conflict detection in accordance with the UIC406 leaflet.

Figure 3.11: Runtime calculation results for a calculated train schedule showing maximum line speeds, tight running and eco-running together with gradient information within a s/v-diagram

3.10 Conflict detection

Conflicts are detected between two objects which are scheduled to use an infrastructure resource at the same time. Besides these conflicts, TPS supports an automated validation of objects for checking the conformance of the object with respect to:

- the allocated infrastructure; or
- operational or planning rules (e.g., controlled meetings, forbidden meetings, minimum headways, rules for violation of train connections , allowances); or
- self-consistency of the data (e.g., checking exclusion rules for specific combinations of train attributes).

The train/train conflicts for concurrent use of infrastructure are detected by applying the rules as set out within the UIC406 leaflet for defining occupation times of infrastructure resources for a train. Consequently, the conflicts between two trains are represented by time overlaps of the trains' occupation times for one and the same resource. The conflicts are shown in the LTG as well as on the Track View (see Figure 3.12 and Figure 3.13). In the tabular timetable editor section, a message box is showing the conflicts handled along with configurable filtering module.

The train/TIR conflicts for concurrent use of infrastructure are detected by checking of overlaps between the train's occupation times and the time for which a TIR of a certain type is imposing an impact on the assigned infrastructure object, e.g., a temporary speed restriction being defined for a subsection of a track. Also these types of conflicts can be depicted from the LTG, the Track View as well as from the conflict message list.

The TIR/TIR conflicts for are detected by checking of overlaps of time for which a two TIRs of certain types are imposing an impact on the assigned infrastructure object. Again, these types of conflicts can be depicted from the LTG, the Track View as well as from the conflict message list.

All conflicts can be filtered and grouped by different criteria, such as e.g. type (headway conflict, crossing conflict, possession conflict), area (station area, line between station areas), extend etc.

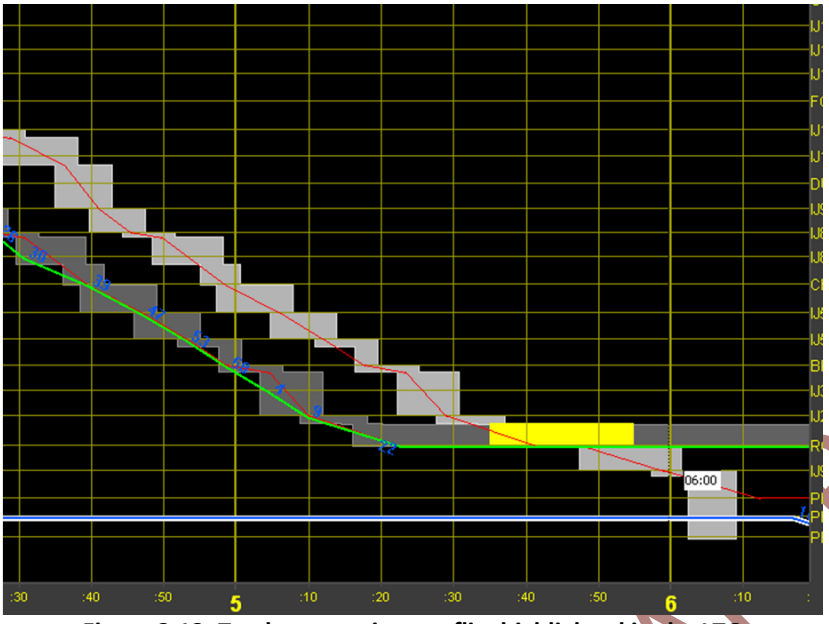

Figure 3.12: Track occupation conflict highlighted in the LTG

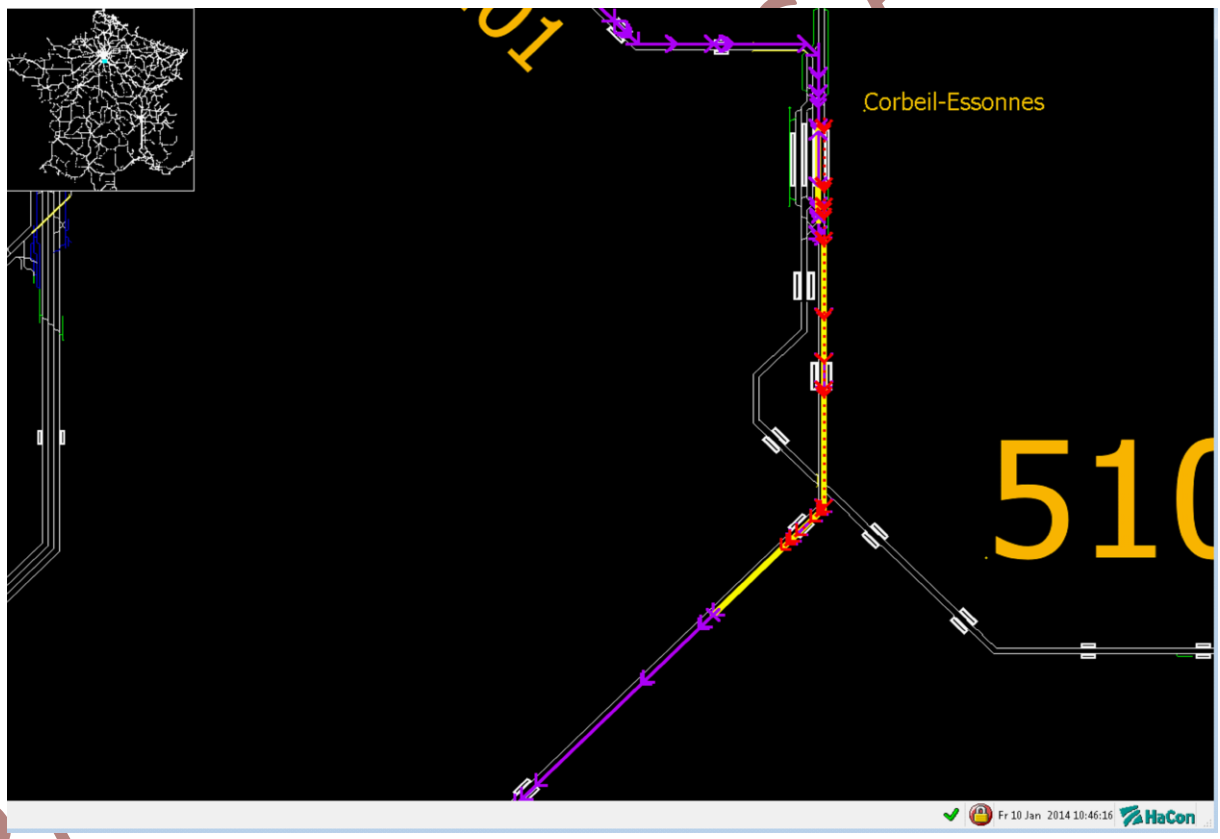

Figure 3.13: Conflict shown on the Track View (yellow highlighted section)

3.11 Conflict resolution

In order to resolve the conflicts, different actions can be taken by the dispatcher. Besides manual adjustments of train schedules or TIR (e.g. possessions), the dispatcher may also invoke Decision Support (DSS) tools as e.g., the automatic conflict resolver "*FindSlot*" or an

optimizer module, both providing decision support to overcome conflict situations by applying offered solutions ad-hoc (see Figure 3.14).

"Decision support" in this context means, that the solutions found by the modules are evaluated and presented in a hitlist ordered by the costs resulting from evaluation. By clicking on an entry, the related solution is applied in a "sandbox" (a private space, not being shared with other users) in order to let the planner study the potential outcome using automated conflict re-assessment, reporting and visualization in the different application components, e.g., the Live Train Graph (LTG).

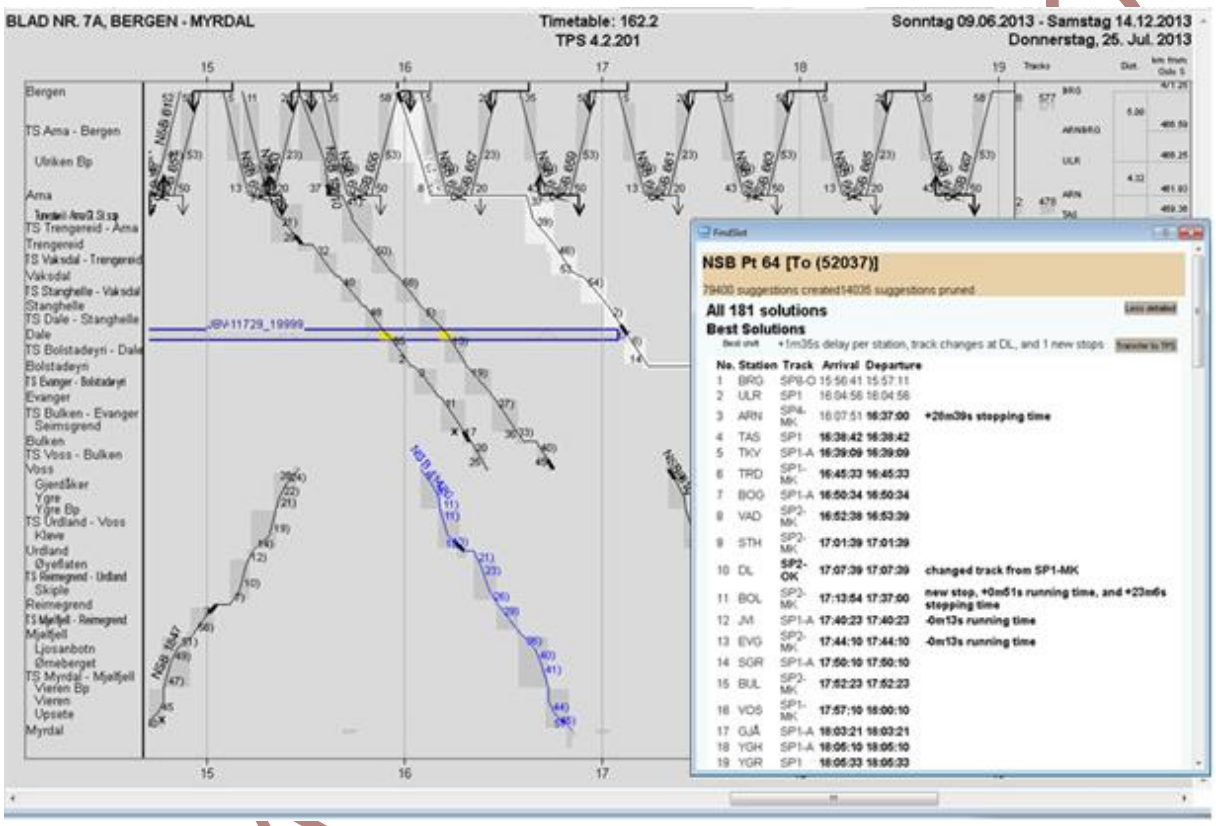

Figure 3.14: Using the "FindSlot" module for conflict resolution (decision support)

FindSlot creates conflict free solutions by combining different optional solution types.

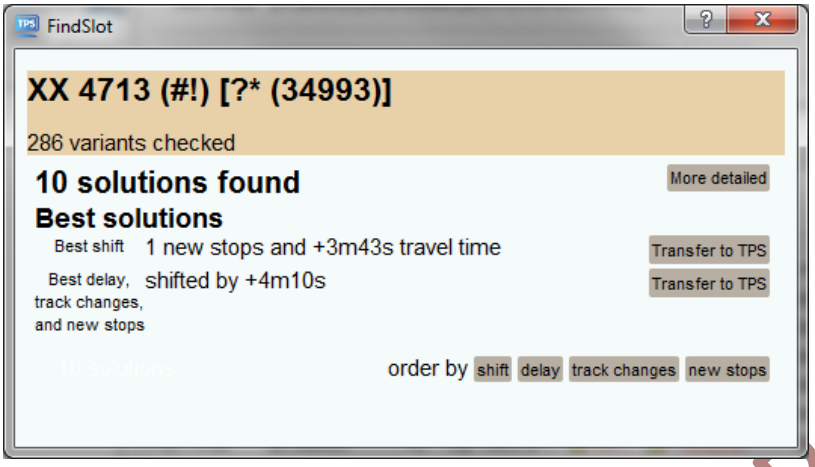

Figure 3.15: FindSlot dialog after retrieval of conflict resolutions as presented to the user

At first after invocation, *FindSlot* shows up with the best solutions within the different possible solution categories (see Figure 3.15). The display can also show more solutions ordered by shift, delay, track changes or number of new stops. Selecting one solution opens up more detailed information about the changes (see Figure 3.16).

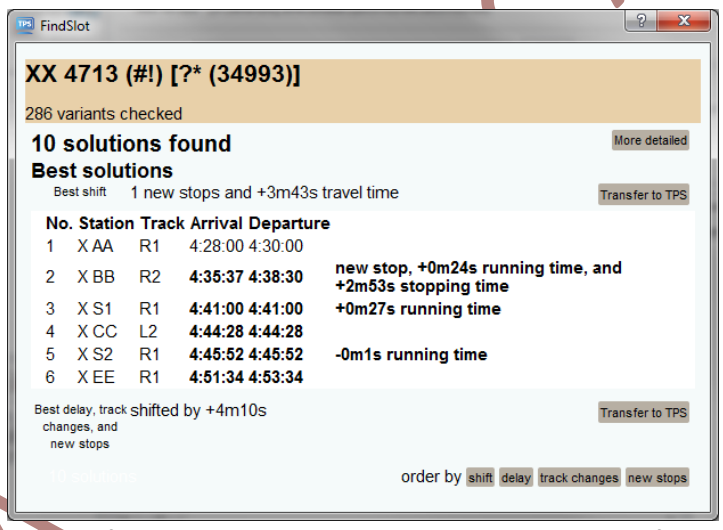

Figure 3.16: More detailed information about the suggested changes resulting from FindSlot module

3.12 PIS System Interfacing (HAFAS)

Exporting published timetable or infrastructure/network restriction information to external systems for passenger information is performed using realtime data enhanced timetable formats. The format being used for the In2Rail demonstrator in Task 7.3 will facilitate the HAFAS data format with enriched realtime data. This format is more or less today's standard for European Train Operating companies and is also used by the UIC for distribution of joint European passenger timetable data.

3.13 Field simulation

The simulation of the field, i.e. simulation of status information of asset objects, trains and other infrastructure elements that comes from the interlocking, RBC and other sources, will be realized by a script which provides the necessary information.

3.14 Order Portal

The order portal is designed as a web portal and used as front end for the capacity allocation process. During this process, external parties create and submit booking requests and agreements for train definitions and receive corresponding offers. The module guides authorised External Entities during the negotiation process. The External Entities are guided through the negotiation using remote access via the internet.

The order portal module consists of a web client, a web server and authentication server. The web front end of the order portal can be loaded by a web browser with a predefined HTTP URL of the web application.

The web server provides the web application to the web browser, i.e. the web application provides the functionality, e.g. to support the process states required for the negotiation process of the train definitions. States are:

- Requested;
- Offered;
- Rejected;
- Agreed;
- Cancelled.

All requests via HTTP are redirected to HTTPS for a secure communication. As the connection is already encrypted, only basic access authentication is used. The authentication server is configured as authentication provider for the web server: thus the authentication server is an external system.

3.15 Import planned TT and Restrictions

An import interface will provide sample timetable and infrastructure/network restriction data from the train planning system TPS to the IMDG.

3.16 Topo Data Import

The Topo Data Import module will translate the input topo description into Canonical model Topology Format. The format to be used is still in discussion and will be decided in WP8.

One possibility could be to use RailML, a data format that is well established in the railway community today, for example in the current version 2.3.

Another way could be the usage of the Rail Topo Model standard that is currently used by WP9 (see Section 6.3.1). Rail Topo Model is a UIC-endorsed topological model for the representation of railway infrastructure. Recently proposed as an international standard (IRS30100), the model has been developed in close cooperation with the railML consortium, and version 3 of the railML standard (currently under development) will serve as the reference "exchange format" implementation for Rail Topo Model within the community.

3.17 Asset Nowcast/Forecast Calculation

The Asset Nowcast/Forecast Calculation module is responsible for the generation of nowcasts and forecasts of the status of the railway assets. It is part of the Asset Nowcast/Forecast chain, depicted in Figure 3.17, which also includes the Asset Nowcast/Forecast View module.

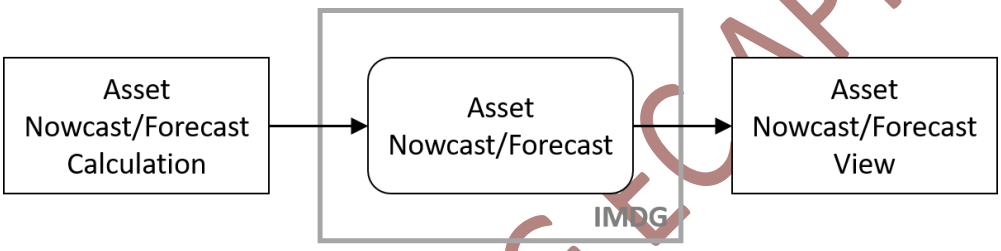

Figure 3.17: Main dataflow for the Asset Nowcast/Forecast group of modules

Firstly, it is important to recall the definition of Nowcasting and Forecasting for the In2Rail project, as defined in Deliverable 9.3 through the work of In2Rail WP9. Forecasting is "the process of exploiting past and present data to make deductions about the future", while Nowcasting can be defined as "the process of exploiting past and present uncertain or incomplete data to make deductions about the present."

Therefore, concerning the Asset Nowcast/Forecast chain of Figure 3.17, the Asset Nowcast/Forecast Calculation module exploits information from various heterogeneous sources in order to produce estimations and predictions of assets statuses. The Asset Nowcast/Forecast View module (described in Section 3.18), instead, is a collection of visualization tools that enable the graphical representation of the Asset Nowcast/Forecast information.

The Asset Nowcast/Forecast Calculation module will integrate and show into the proof-of-concept the achievements of the In2Rail WP9 – Nowcasting and Forecasting, by means of two scenarios proposed by Rete Ferroviaria Italiana (RFI) and by Strukton Rail (SR). The two nowcasting and forecasting scenarios are described in the appendices of this document, in Section 6.1 and 6.2 respectively.

The part of the proof-of-concept related to WP9 will focus on demonstrating two different aspects of the work done:

- The Data-driven Methodology that is exploited in order to propose a solution for all the scenarios defined in the context of WP9;
- The Big Data Architecture, which supports data-driven methodologies so to produce asset status nowcasts and forecasts from large amounts of heterogeneous data, which can be collected from any data source.

The Asset Nowcast/Forecast Calculation module will comprise both these two different aspects, which are described in the following subsections.

3.17.1 Data-driven Methodology

The need for extracting the valuable information content hidden into data and transforming it into actionable knowledge gave birth to and sanctioned the rise of data-driven methodologies and techniques, developed by cross-relating theories coming from several different fields (such as mathematics, statistics and numerical optimization). Data-driven methodologies refer to the exploitation of statistical techniques for the numerical elaboration of historical observations of inputs and outputs of a specific system, behaviour or phenomenon, aiming at building models that can then be used to estimate the current outputs or to predict the future ones. These methods do not require any prior knowledge of the physical system, since the resulting model is usually not supported by any physical interpretation. Moreover, since the model is inferred based on measurements affected by different types of noise, this process is intrinsically under the effect of uncertainty and therefore a significant amount of data are required for building reliable models.

Figure 3.18 shows the (simplified) set of steps needed to generate a data-driven model able to respond to real-time inputs. In this proof-of-concept, the process is performed inside the Big Data Architecture (described in the next subsection).

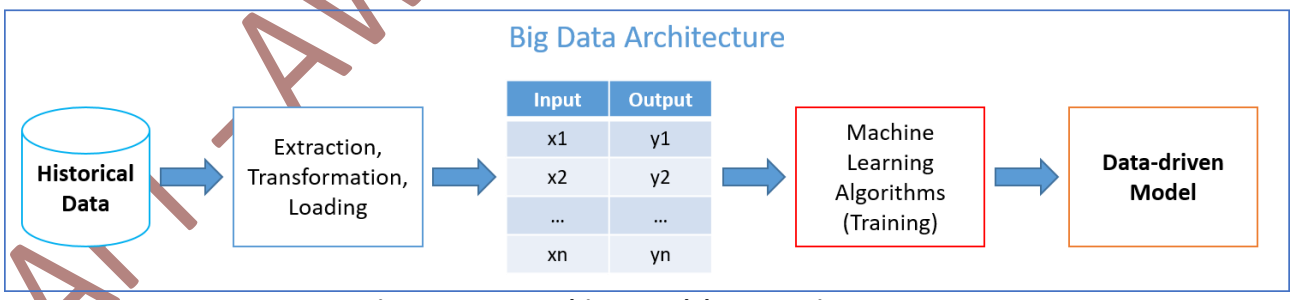

Figure 3.18: Data-driven Models Generation

In particular, the first step deals with the management of problems and inconsistencies included in the set of historical raw data (statistically representative of the system, behavior or phenomenon under examination) to be analyzed. The raw data formats are “railML” – for timetable, infrastructure, rolling stock and other types of information applicable to the standard – and “sensorML” – for components sensor related measurements and

information/metadata. These two data formats (described in Section 6.3.1 and Section 6.3.1.1 respectively) are integrated at a semantic level through the usage of ontologies and semantic mediation. Therefore, the raw data is Extracted, Transformed and Loaded (ETL) so to create a dataset composed of pairs of inputs and outputs that can be processed by machine learning algorithms.

The next step is called "training", since machine learning algorithms look at the data and try to reproduce the relationship between inputs and outputs of the dataset. These algorithms perform a large number of multiple iterations on data in order to refine the solution with a "step by step" philosophy. Therefore, the underling architecture supporting machine learning algorithms has to be extremely powerful and reactive to allow completing this step in a feasible and actionable time frame.

Finally, the result of this process is a data-driven model that can be used to respond to new, previously unseen real-time input data to produce estimates or predictions of real-time outputs, as shown in Figure 3.19.

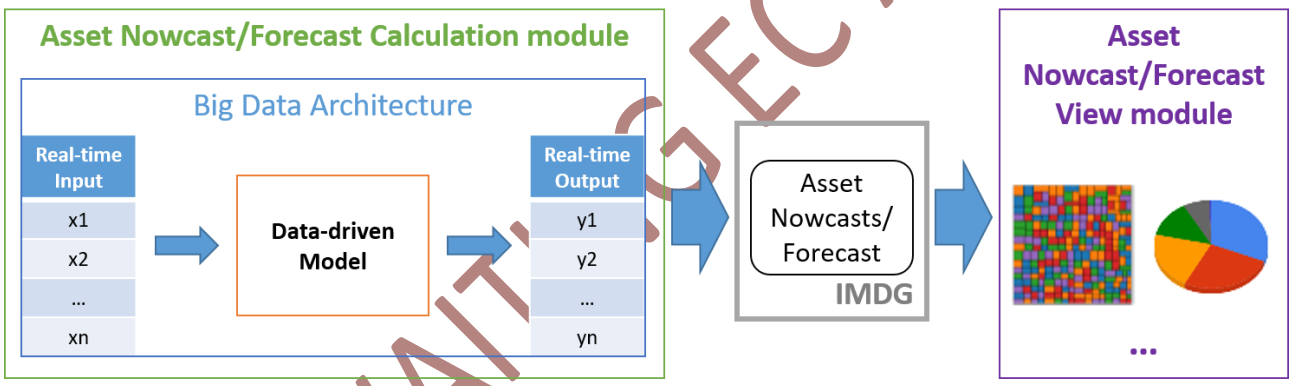

Figure 3.19: Real-time application of data-driven models

It is worth noting that the real-time output, calculated in real-time inside the Big Data Architecture, is pushed firstly to the IMDG, which makes it available to the Asset Nowcast/Forecast View module for visualization.

3.17.2 Big Data Architecture

A Big Data Architecture (BDA) is a combination of hardware and software technologies aiming at storing, processing, analyzing and visualizing large amounts of data (either structured or unstructured) in order to extract the information content hidden inside it. In this proof-of-concept, a prototype of a BDA will be developed and exploited, so to demonstrate the achievements of In2Rail WP9 – Nowcasting and Forecasting. The concepts described in this section are directly derived from the "Deliverable 9.2 – Data/Information Management System architecture design" of the WP9 of the In2Rail project.

From a hardware point of view, a BDA is composed of a cluster of commercial off-the-shelf (COTS) servers, which guarantee performance and redundancy of the information system

(see Figure 3.20). It includes a few frontend nodes that allow accessing, using and monitoring a cluster of interconnected data nodes, which provide both storage and computing power in the same boxes. The main advantages of this type of architecture are high scalability (new data nodes can be added online, expanding both storage and computing capabilities), availability (hardware and software redundancies ensure that the system is always on and that no data is lost) and flexibility (mixed types of hardware and software components can work together effectively).

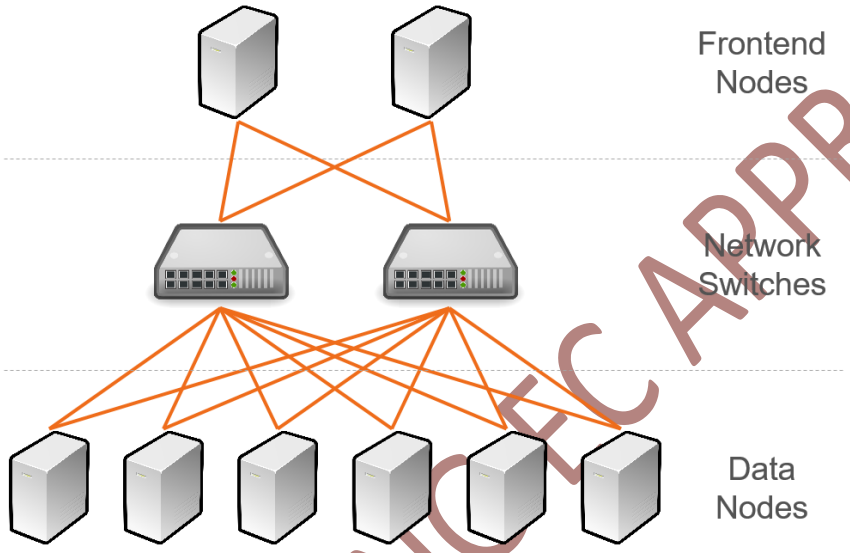

Figure 3.20: Big Data Architecture - Hardware perspective

From a software point of view, Figure 3.21 shows the generic logical schema on which modern BDAs are based. The typical BDA considered in the context of WP9, and therefore exploited for the In2Rail proof-of-concept, is based on the Apache Hadoop ecosystem. Since the proof-of-concept focuses on showing the potentialities of the methodologies and technologies selected and developed in the context of WP9, only the core parts of a BDA will be considered and consequently described in the next subsections. It is worth noting that the Data Visualization logical block of Figure 3.21 represents the set of interfaces that will allow the outputs of the Asset Nowcast/Forecast Calculation module (i.e. of the data-driven models) to be exchanged with the Asset Nowcast/Forecast View module through the IMDG.

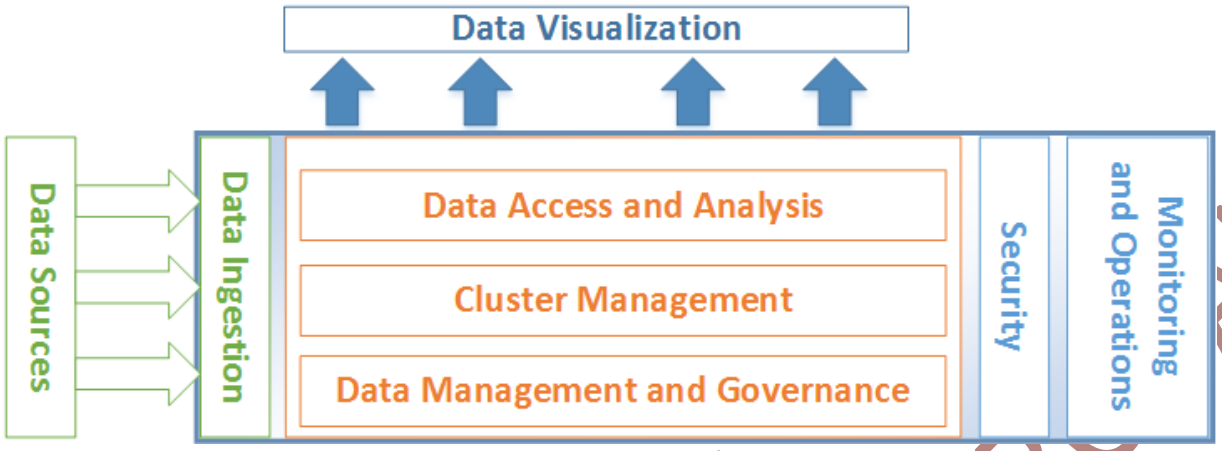

Figure 3.21: Big Data Architecture – Software perspective

3.17.2.1 Big Data Management and Governance

Big Data Management refers to the possibility of storing any kind of data, both structured and unstructured, into the storage system of the BDA. The core elements of this logical block are Distributed File Systems (DFS). This kind of file systems allow storing large amounts of data in a distributed fashion by taking advantage of the capacity included in a cluster of computers, showing them as a single entity, i.e. grouping the single capacity of each node of the cluster into a unique storage system. The computer cluster can contain specialized hardware for data storage as well as commodity hardware: all the storage will behave in the same way, looking as a unique huge disk, where the complexities of the cluster architecture have been transparently hidden. In order to avoid data losses, specialized software mechanisms ensure that data is redundantly stored in several nodes in the cluster. Data replication mechanisms usually improve the cluster management and the overall performances of DFSs. Moreover, DFSs are highly flexible and scalable, indeed new storage devices or data nodes can be added or removed from the cluster easily and seamlessly.

Hadoop Distributed File System (HDFS) is an open source distributed file system designed to run on commodity hardware, and represents the selected DFS for implementing the In2Rail proof-of-concept. Although it has many similarities with existing distributed file systems, the differences from others are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware, provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. It is designed to address hardware failures by providing data duplication and fault tolerance mechanisms. Since it is designed more for batch processing rather than interactive use, it allows for streaming data access, emphasising on high throughput of data access rather than low latency of data access. HDFS implements a write-once-read-many access model for files, which means that once a file is created, written, and closed, it cannot be changed. This assumption simplifies data coherency issues and enables high throughput data access.

3.17.2.2 Big Data Cluster Management

Big Data Cluster Managers are fundamental operational tools of a BDA that allow looking at the computer cluster as a single computing entity sharing the computing resources of all the nodes. These systems are responsible for resource management and job scheduling, monitoring and recovery (in case of failures). The former functionality takes responsibility for managing any computing component (CPU, memory, disk and network) of every node of the cluster, which is considered as a sort of "resources container". Obviously, the usage of every component has to be recorded and reported.

YARN (Yet Another Resource Negotiator) is the selected Big Data Cluster Manager for implementing the In2Rail WP9 proof-of-concept. It represents the architectural centre of Hadoop that enables simultaneous data processing in multiple ways. In particular, it provides:

- A framework for writing data access applications that run in Hadoop;
- Resource management;
- Pluggable architecture for a versatile range of processing engines that enable to interact with the same data in multiple ways, at the same time, with the best available tool (batch, interactive SQL, NoSQL, etc.).

3.17.2.3 Data Access and Analysis

Big Data Access and Analysis refers to all the tools developed to manipulate data, to implement efficient data storage and retrieval through database management systems, to analyse data and generally to perform any kind of computation involving data inside the BDA. In particular, Big Data Access relates to two different categories of tools: tools for data manipulation, implementing one or more processing modes, and distributed DataBase Management Systems (DBMS), which support different storage needs and consequently allow for different data access patterns.

Among the vast availability of open source tools, Apache Spark has been selected for implementing the In2Rail WP9 proof-of-concept. It represents a fast in-memory data processing engine (with elegant and expressive development APIs) that allow data workers to execute efficiently streaming, machine learning or SQL workloads requiring fast iterative access to datasets. The Hadoop YARN-based architecture provides the foundation that enables Spark and other applications to share a common cluster and dataset while ensuring consistent levels of service and response. Apache Spark consists of Spark Core and a set of libraries. The core is the distributed execution engine and the Java, Scala, and Python APIs offer a platform for distributed ETL application development. Additional libraries, built atop the core, allow diverse workloads for streaming, SQL, and machine learning.

3.17.2.4 Data Ingestion

Data Ingestion tools allow loading and managing data according to the required policies, coping with the huge volume, high velocity, or variety of the data and the related sources. These tools are able to interface with several heterogeneous sources, such as data streams, RDBMS and traditional EDW, messaging interfaces, other distributed filesystems, and the like. Moreover, they include functionalities to validate, cleanse, transform, reduce, and load the data into the big data lake for further processing. In other words, Big Data Ingestion represents the set of interfaces of the BDA with the external word, both inward and outward.

Data Ingestion tools can only be successfully developed if they are tightly coupled with the proposed BDA, which not only means that every "connector" needs to have a proper counterpart internally to the BDA, but also that they have to be able to exploit the distributed nature of this kind of systems. Indeed, great improvements can come if the ingestion tools are able to work simultaneously from different working nodes in the cluster, virtually eliminating any bottleneck with the external world.

The right tool for demonstrating the ingestion of data in the context of the In2Rail proof-of-concept is selected among the set of open source tools available to Hadoop-based BDA.

3.18 Asset Nowcast/Forecast View

The Asset Nowcast/Forecast View module is the collection of different tools that allow presenting the Asset Nowcasts/Forecasts (i.e. the results of the Asset Nowcast/Forecast Calculation module) in a pictorial or graphical format. It will enable demonstrating that decision makers could take advantage of visual analytics information so to grasp difficult concepts or identify patterns in data. Visualization has proven effective for not only presenting essential information in vast amounts of data but also driving complex analyses. For these reasons, tools for Asset Nowcast/Forecast View are extremely important.

Within the PoC, different Tools will be chosen to outline in the best and most comprehensive way the results achieved from Big Data analysis.

3.19 ATO Export

The ATO Export module will convert a forecast timetable, which is formatted in the canonical model, into an ATO Journey Profiles. Figure 3.22 shows graphically the dataflow from forecast timetable to ATO Journey Profiles.

Figure 3.22: ATO Export dataflow

The conversion will be evaluated based on current works in X2Rail-1 WP3 project with the specification of Track-Train Interface for ATO over ETCS (Subset 126). The most important aspect to be evaluated is if the Canonical Model provides all required data for calculation of Journey Profiles.

4 Conclusions and Outlook

To summarize the key aspects for the proof of concept Table 4.1 provides an overview of the actions that have to be taken and the results needed to verify the suitability of the WP8 architecture towards the UseCases defined in 7.2 via the approach described in chapter 2.

ASPECT	APPROACH	EXPECTED RESULTS
Data Storage and distribution Layer (see chapter 3.1)	Since there are multiple software tools available that implement IMDG, at least 2 concrete products shall be tested.	The In²Rail architecture shall perform on 2 of e.g.: • HazelCast; • Redis; • InfiniSpan; • Other.
Data Grid Wrapper (see chapter 3.2)	The wrapper shall be implemented in 2 different programming languages and for at least 2 different IMDGs	Two out of: • HazelCast and Java; • Redis and C++; • HazelCast and C++; • InfiniSpan and Java; • Etc.
Data Grid Explorer (see chapter 3.3)	A software tool shall be created to browse the IMDG and visualize the data objects. It shall have a function to check consistency of data structures	Software tool
Functional modules and tools (see chapter 3.4 ff.)	These toolboxes shall be implemented to work with the data inside the IMDG and to proof, that a suitable operation for each of the use cases is possible with this architecture.	Software tools to manipulate data inside the IMDG. These tools shall solve the Use Cases in 7.2
Canonical data model	The data model needs to be defined for the software tools. It is important to have a procedure to migrate this model into the key-value format for the IMDG	Canonical data model description. Translation method/procedure. BLOB serialization
Overall	The idea is to create concrete software modules that implement the abstract “Integration Layer” concept with real world objects. With this software, selected use cases will be tested under conditions we expect in the reality.	• Qualitative toolchain for the use cases. • Quantitative measurements to judge system performance, e.g. scalability, fail operation.

Table 4.1: Overview of planned actions in In2Rail proof-of-concept

One key aspect to proof the architecture and the overall approach of the IL, it is necessary to provide deeper insight in how the described canonical data model will look like. It is a

mandatory prerequisite to define this data model in such a way, that it can be translated into the key-value pairs representation of IMDG in a deterministic way (see chapter 3.2). Meaning, the translation of data objects used by the applications shall allow a 1:1 rule to migrate the data object names to IMDG addresses (keys).

To enable cooperation between different tools coming from different vendors the payload shall be encoded in a way specified by WP8. In the context of the proof-of-concept the effort for payload creation and efficiency for payload distribution and parsing shall be evaluated.

The final Deliverable of the In2Rail proof-of-concept, D7.5 will show the results of the proof-of-concept and therefore validate if the requirements in [In2RailL D7.2] can be fulfilled with the selected solutions.

5 References

[In2Rail D7.1] In2Rail Project, Grant Agreement H2020-MG-2014-635900 – **D7.1 State-of-the-Art and High Level Requirements –** 30/09/2015, http://www.in2rail.eu/
[In2Rail D7.2] In2Rail Project, Grant Agreement H2020-MG-2014-635900 – **D7.2 I2M Consolidated functional and non-functional requirements –** 30/04/2016, http://www.in2rail.eu/

6 Appendices

6.1 WP9 Scenario by RFI

Title	**Data-driven Train Delay Prediction based on heterogeneous data coming both from railways and external sources**
Organisations Involved	• Rete Ferroviaria Italiana **(RFI)** • University of Genoa **(UNIGE)**
Objective(s) of the scenario	The main objective of this scenario is to nowcast and forecast train delays, based on historical information related to train movements, on the historical planned timetable, and on historical weather conditions data. "Train movements" are records of timestamps of arrivals (departures) of trains at (from) a particular "checkpoint", which represents any location where the position of a train associated with a timestamp is recorded. The "planned timetable", instead, can be seen as a table including timestamps of scheduled arrivals (departures) for each train and for each checkpoint included the journeys of the trains. Finally, "weather conditions" data includes several atmospheric variables such as temperature, humidity, wind speed and direction, and the like. In particular, the scenario develops around the possibility of integrating into predictive models for train delays some exogenous information with respect to the railway world (i.e. weather conditions). The scenario aims at designing, implementing and testing in laboratory (and validating at a future stage) a set of data-driven predictive models for nowcasting and forecasting purposes based on data related to the Italian railway network provided by RFI.
Relationship with TMS and/or maintenance	By providing accurate train delay predictions to TMSs, it is theoretically possible to improve traffic management and dispatching in terms of: • Passenger information systems, increasing the perception of the reliability of train passenger services and, in case of service disruptions, providing valid alternatives to passengers looking for the best train connections. • Freight tracking systems, estimating goods' time to arrival correctly so to improve customers' decision-making processes. • Timetable planning, providing the possibility of updating the train trip scheduling to cope with recurrent delays. • Delay management (rescheduling), allowing traffic managers to reroute trains so to utilize the railway network in a better way.
Description of the scenario	This scenario aims at developing Nowcasting and Forecasting models able to estimate the current delay of a train and to predict it at

	future times. In other words, the main goal is to predict the series of delays that will affect a specific train in all the subsequent "checkpoints" included in its journey, with the highest possible accuracy and, when possible, with an estimate of the forecasting accuracy itself. A large literature covering this problem already exists, considering the railway network parameters and the information that can be retrieved from the classical railway information systems as the only data sources (e.g. the TMS for records about train movements). However, many other exogenous factors affect railway operations, such as passenger flows, strikes, celebrations and similar public events. For this reason, this scenario concentrates on nowcasting and forecasting train delays by means of a set of data-driven models integrating data about past train movements and weather conditions. In particular, by combining train movements data and weather data into a single historical database, machine learning algorithms will analyse the data so to build data-driven models that are able to respond to new, previously unseen input data to predict train delays.
Data exploited for the scenario	**Railway Data** From every TMS, it is possible to retrieve data about train movements, with time and position references (e.g. timestamps at station arrivals and station unique IDs), and theoretical timetables, including planning of exceptional train movements, cancellations, etc. UNIGE received from RFI data related to two of the main areas of the Italian railway network about 2015, spanning over one year for the first area and 6 months for the second one. Note that the information have been anonymized for privacy and other concerns. **Weather Conditions Data** The information about weather conditions have been retrieved from weather stations in the area of interest. The association between train movements data and weather data has been performed by looking for the closest weather station to the railway station/line/checkpoints. The values included in the data are related to atmospheric pressure, solar radiation, temperature, humidity, wind direction and speed, and rainfall. Note that UNIGE downloaded weather data related to the same areas and to the same time intervals for which we have train movements data provided by RFI.

Table 6.1: Tabular Description for Scenario by RFI

Figure 6.1 shows a pictorial view of the scenario, focusing on expressing the idea that the main goal is to predict the series of delays that will affect a specific train in all the subsequent "checkpoints" included in its journey.

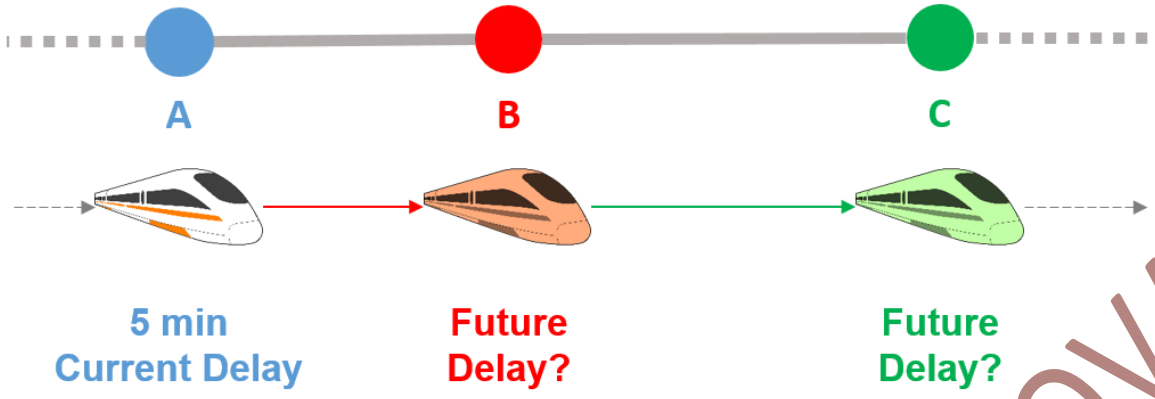

Figure 6.1: Pictorial representation of RFI scenario

6.2 WP9 Scenario by SR/UNIGE

Title	**Datamining correlation and influence of maintenance/repair actions and weather conditions on railway assets**
Organisations Involved	• Strukton Rail Netherlands (SR) • University of Genoa (UNIGE)
Objective(s) of the scenario	The main objective of this scenario is to forecast possible failures of assets based on the correlation of past asset failures and past weather conditions or maintenance actions, in particular considering a set of different infrastructure assets selected as the most relevant ones from the TMS perspective (see Deliverable 9.1 of the In2Rail project). Therefore, this scenario aims at designing, implementing, testing and, at a future stage, validating a set of predictive models for forecasting purposes, based on data provided by SR about maintenance/repair actions, weather and failures/malfunctions. The problems that will be investigated (and for which one or more forecasting models will be developed) are: 1. Correlation and influence of executed maintenance/repair actions on failures/malfunctions. 2. Correlation and influence of the weather conditions on failures/malfunctions.
Relationship with TMS and/or maintenance	The information outputted by the models can be very useful for a TMS because it could be used by the TMS to reroute trains through safer paths, minimizing the risks of any problem. Moreover, the same output could be used by the maintenance department in order to schedule proper maintenance actions that could prevent additional or worst problems.
Description of the scenario	Every time an infrastructure asset is affected by a failure/malfunction, it is clear that this will affect not only the single asset functional behaviour, but also the normal execution of railway operations. The functional behaviour of railway infrastructure assets degrades for many different reasons: age, extreme weather conditions, heavy loads, and the like. Additionally, problems can be

	introduced unknowingly by performing maintenance actions, for example by a simple human error or as a reaction of the system to changes made on an object. For these reasons, this scenario aims at investigating two among all the factors that might affect the degradation of assets, i.e. maintenance/repair actions and weather conditions, and at designing and developing new forecasting methodologies by exploiting data-driven predictive techniques. A set of predictive models able to forecast the probability of failures for a particular asset will be developed based on the data provided by SR about historical weather conditions, performed maintenance/repair actions and failures/malfunctions.
Data exploited for the scenario	**Weather condition:** data retrieved from the Royal Netherlands Meteorological Institute (KNMI) **Maintenance/repair actions:** historical datasets regarding the maintenance/repair activities (including their duration) will be provided by Strukton Rail. This data is collected by Strukton Rail but commissioned by ProRail (Dutch rail infrastructure manager). Data is originally stored in a Maintenance Management System. This information could be provided by Asset Manager, ProRail. **Failures/Malfunctions:** historical datasets regarding the recorded failures/malfunctions will be provided by Strukton Rail.

Table 6.2: Tabular Description for Scenario by SR/UNIGE (1)

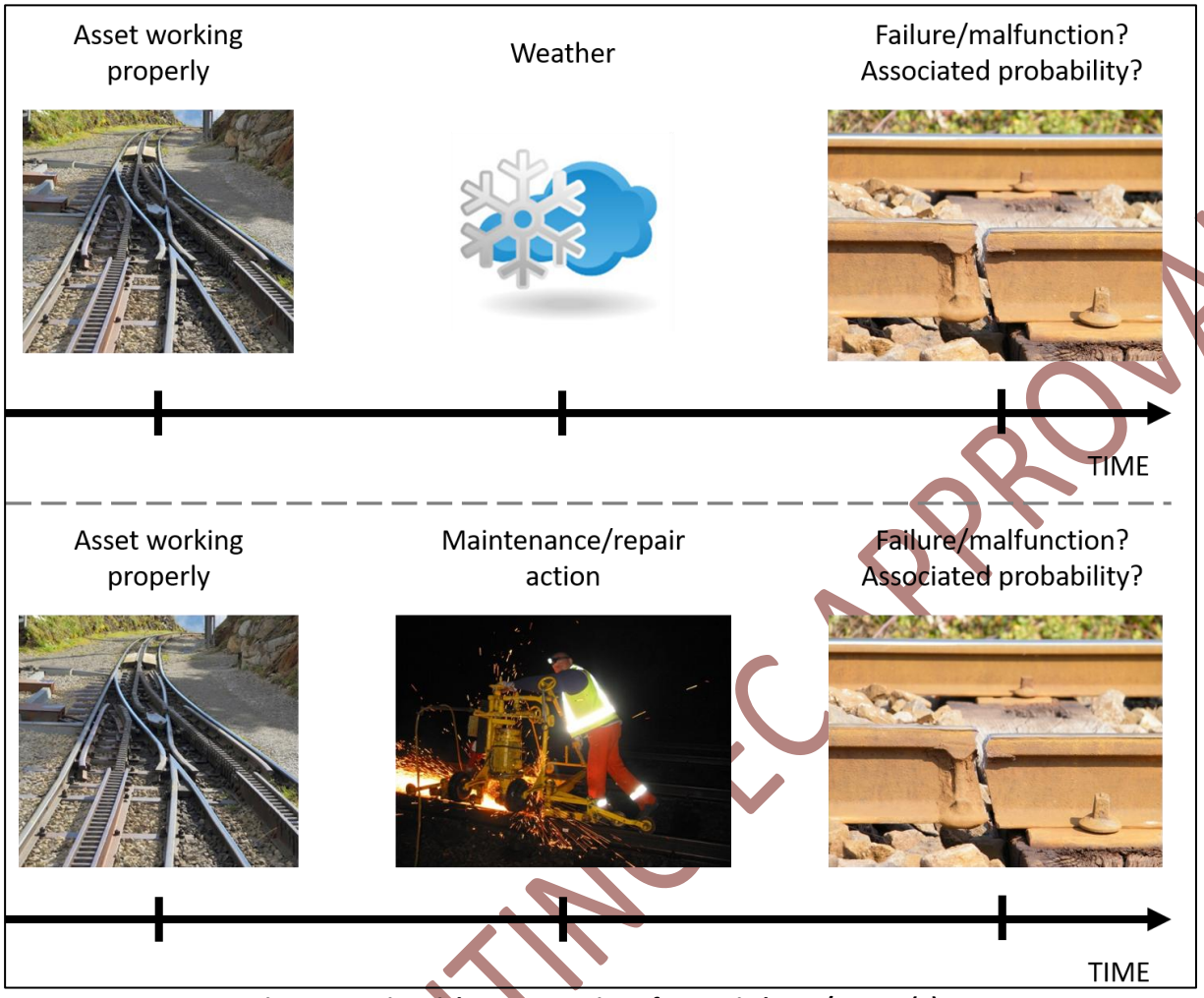

Figure 6.2: Pictorial representation of scenario by SR/UNIGE (1)

Title	**Analysis, estimation and prediction of time to restoration for different assets and different failures/malfunctions based on maintenance/repair reports**
Organisations Involved	• Strukton Rail Netherlands (SR) • University of Genoa (UNIGE)
Objective(s) of the scenario	An analysis will be carried out about the "time to restoration" (or "repair time") needed to restore the asset to a proper functional state after a specific failure/malfunction occurred. The analysis will try to develop a forecasting methodology able to estimate in advance the precise repair time once a problem on an infrastructure asset arises. Therefore, this scenario aims at designing, implementing, testing and, at a future stage, validating a set of predictive models for forecasting purposes, based on data provided by SR about maintenance/repair actions and weather data.
Relationship with TMS and/or	Since in this case the output of data-driven models is the time required to complete a repair action, it could be used by the

maintenance	maintenance department in order to schedule proper maintenance actions. As a positive byproduct, the maintenance department could communicate better estimations of the time to restoration to the TMS, which could exploit this information for planning and managing line possessions in an informed way.
Description of the scenario	Every time an infrastructure asset is affected by a failure/malfunction, it is clear that this will affect not only the single asset functional behaviour, but also the normal execution of railway operations. For this reason, the objective of the third analysis is to estimate the time to restoration for future (planned) and urgent maintenance actions by looking at the past maintenance reports, correlated to the different assets and different types of malfunctions. The predictive models that will be designed will be able to exploit the knowledge enclosed into maintenance reports so to predict the time needed to complete a maintenance action over an asset in order to restore its functional status. Moreover, historical weather conditions data will be included in the analysis in order to take into account of the atmospheric factors affecting railway maintenance/repair operations (e.g. fog reducing visibility).
Data exploited for the scenario	**Weather condition:** data retrieved from the Royal Netherlands Meteorological Institute (KNMI) **Maintenance/repair actions:** historical datasets regarding the maintenance/repair activities (including their duration) will be provided by Strukton Rail. This data is collected by Strukton Rail but commissioned by ProRail (Dutch rail infrastructure manager). Data is originally stored in a Maintenance Management System. This information could be provided by Asset Manager, ProRail. **Failures/Malfunctions:** historical datasets regarding the recorded failures/malfunctions will be provided by Strukton Rail.

Table 6.3: Tabular Description for Scenario by SR/UNIGE (2)

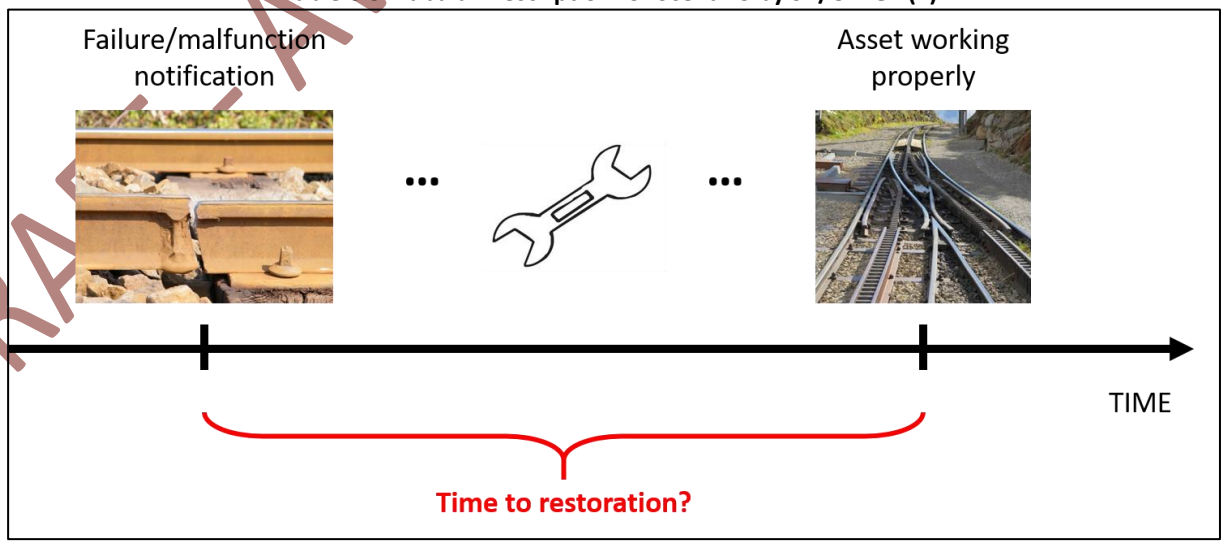

Figure 6.3: Pictorial representation of scenario by SR/UNIGE (2)

6.3 WP9 Data Formats

The WP9 data formats are “railML” – for timetable, infrastructure, rolling stock and other types of information applicable to the standard – and “sensorML” – for components sensor related measurements and information/metadata. These two data formats (described in Section 0 and Section 6.3.1.1 respectively) are integrated at a semantic level through the usage of ontologies and semantic mediation.

6.3.1 railML

The original railML models (railML, railML 2) are a series of XML schemas produced by a loose consortium of railway companies, academic institutions, and consultancy firms (see www.railml.org for further information)

The models are designed to capture four, largely static, elements of the railway and railway operations:

- Common concepts;
- Timetable;
- Rolling stock;
- Infrastructure (at the macroscopic and microscopic levels).

Alongside the network topology, the railML infrastructure model can also contain information on the locations and equipment types of key operational & control assets, e.g. signals, balises, axel counters and level crossings, as well as the presence of linear assets such as electrification equipment. However, while the model can describe the presence of these assets, it does not capture their dynamic state (e.g. the lie of a particular switch, or the presence of an obstruction on a crossing). Operational parameters, such as speed limits, can also be captured.

6.3.1.1 sensorML

The SensorML model is designed to allow the description of the processes of data collection and transformation, including the description of any sensors and actuators that are involved in the process. From a railway perspective, this can be thought of as a description of a crossing barrier, sensors attached to the asset such as a current clamp on the barrier motor, and the processing performed on the current waveform such as down sampling of the data and baseline adjustment. As with many XML-based standards, SensorML includes a certain amount of flexibility in terms of the information that must be included in a valid file, and as a result can be expressed in a very compact form if required, although this comes at the loss of contextual information about the data.

One very convenient feature of the SensorML model is that it includes native support for describing sensor data that is accessed via resources remote to the file, via web services or

similar), this provides a useful way of dividing raw sensor data values from the description of the sensor system that is generating them, and hence (with an appropriate choice of encodings) allows very effective use of bandwidth once the system configuration itself has been described.

For further information on sensorML visit for example www.sensorml.com or http://www.opengeospatial.org/standards/sensorml