Information Systems 49 (2015) 126–141

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Efficient formalization of railway interlocking data in RailML

Mark Bosschaart [a], Egidio Quaglietta [a,*], Bob Janssen [b], Rob M.P. Goverde [a]

[a] Delft University of Technology, Faculty of Civil Engineering and Geosciences, 2628 CN Delft, The Netherlands
[b] Siemens NL, Department of Rail Automation, Prinses Beatrixlaan 800 2595 BN Den Haag, The Netherlands

ARTICLE INFO

Article history:
Received 30 October 2014
Received in revised form
17 November 2014
Accepted 18 November 2014
Available online 27 November 2014

Keywords:
Railway interlocking
RailML database
UML class diagram
Simplification of railway engineering processes
Knowledge-based railway systems

ABSTRACT

Railways wish to reduce the costs of engineering interlocking systems by simplifying the exchange of technical information between stakeholders. Exchanging interlocking data in a machine-readable and standardized format removes inefficiency due to misinterpretation and conversion of data from non-standard, often paper-based formats. This paper proposes an UML class representation of the interlocking data that maps the characteristics of the interlocking system. These data model represents both tabular and geographical interlocking. The model reflects the topology of the railway network that the interlocking controls, routes, relations between signal aspects, speed indication signals, automatic train protection (ATP) and the state of movable and non-movable track elements. The data are machine-readable RailML, an incarnation of XML that is gaining the status of standard in railway modelling. We applied our approach to model the Dutch Santpoort Noord station, starting from paper-based track and signal plans. The resulting database captures all the features of the interlocking, whilst removing data redundancy. The model has general validity and it is easily extendible to other interlocking and ATP systems. Our approach contributes to create standard, electronic interlocking databases, directly from technical documents currently used in the practical world.

1. Introduction

Safety of train operations is the first requirement of signalling and interlocking. Fixed block signalling systems maintain trains at a safe distance by enforcing that only one single train occupies a block section between two consecutive signals. When a block section is occupied, the signalling system forces the following trains to slow down at restrictive signal aspects (e.g. yellow) until they stop at the red signal without entering the occupied block. An Automatic Train Protection (ATP) system warns and/or applies emergency brakes if a train overcomes speed indications (ATP speed indications) or threatens to overrun a signal at danger. ATP continuously or intermittently checks that train speeds are within the limits of the ATP speed indications. Signalling and ATP guarantee safe train movements on open track between stations. In stations or junctions, we need an interlocking system that sets signals, switches and prevents train collisions. An interlocking will move and lock track elements such as level crossings and/or switches in lock step with signals creating a *route* through the network that a given train can safely travel. A *route* is therefore a sequence of infrastructure elements, track detection sections, movable track elements (e.g. switches) and signals that are set and locked for a train to travel from one signal to the next. Calling a route between two signals therefore means: *i*) reserving a sequence of track detection sections for the passage of the train, *ii*) locking switches in a direction (i.e. straight, right/left) such that our train can move safely whilst other trains cannot enter that route, and *iii*) setting the proper signal aspects. The main task of the interlocking is to call and lock disentangled routes through a network

* Corresponding author. Tel.: +31 152782761; fax: +31 15283179.
E-mail address: e.quaglietta@tudelft.nl (E. Quaglietta).

http://dx.doi.org/10.1016/j.is.2014.11.007
0306-4379/

preventing collisions between trains. Signal aspects depend on the condition of other infrastructure elements, i.e. the direction of switches, the occupancy of track detection sections and the speed indications on the track. Typically signals assume restricted aspects when a train approaches deviating switches, signals at danger or tracks with restricted speed. To ensure safety there are formal dependencies among signal aspects, switch directions, ATP speed indications and occupancy of track detection sections, that must be respected. An interlocking must be aware of all these dependencies in order to exclude the calling of conflicting routes. The entire set of these dependencies defines the logic of an interlocking system.

The logic of the interlocking system can differ depending on the technology adopted, i.e. electric, mechanical, electronic, as well as country-specific rules imposed by the signalling and ATP systems. Beside the logic, the topology of the network, i.e. the mutual position of track sections, switches and signals, determines the engineering of an interlocking.

Collecting the rules and data for designing and engineering[1] an interlocking is demanding. The various design and engineering phases involve a complex loop of human interactions between Infrastructure Managers (IM), i.e. the owners of the railway infrastructure, and Industrial Suppliers (IS). The latter tailor the interlocking system to fit the particular railway station, meet technical requirements and contract specifications.

The loops between IMs and ISs consume much time, particularly for engineering and testing, because information about the logical and topological interlocking characteristics is conveyed on paper plans or files that are neither machine-readable nor in a standard format. Both IMs and ISs must interpret the plans and/or the files, manually extract the data, and convert them to a specific format readable by their engineering tools. Such data exchange is clearly inefficient and error-prone, protracting the engineering phase. The probability of error is proportional to the size of the network that the interlocking will control. Errors can mean lack of safety, hence potential loss of life. This means that thorough and complete testing must find any error. The cost of correcting errors increases approximately by a factor of 10 with each step of design, factory testing, and on-site testing.

In such a context, there is the urgent need for both IMs and ISs to reduce the complexity of the data exchange process by describing and collecting interlocking data in a format that is machine readable, standard, and last but surely not least correct. Data for interlocking can be exchanged and processed in an automated and efficient process that strongly reduces errors, time and costs during engineering and test phases. Literature proposes different approaches (e.g. [13,14] to formalize interlocking data into a standard and machine-readable format. The main drawbacks with these formalizations are: *i*) they are case-specific and do not generally apply to different kinds of interlocking; *ii*) they do not describe the interlocking dependencies among signal aspect, speed indications and directions of movable elements.

[1] Design is the building and programming of *generic* interlocking hard- and software. Engineering is the process of *tailoring* a generic interlocking product to a *specific* railway station.

In this paper we overcome these limits by introducing a formalization of the interlocking data that is generally valid and describes specific interlocking dependencies. The formalization is inspired by schematic plans used by IM's and interlocking providers to graphically represent interlocking relations. We expressed the model in terms of RailML (Railway Markup Language) [12,15], a dialect of the extensible markup language (XML) which is increasingly being used to exchange standardized data between different railway software packages.

Section 2 of this paper describes the interlocking system and its characteristics. Section 3 reviews literature about mathematical and descriptive models of interlocking systems. Section 4 describes the RailML language. Section 5 gives a view on current approaches used to engineering interlocking systems. Section 6 puts forward our novel interlocking formalization. A practical application of the approach is reported in Section 7 where we formalize the Dutch station Santpoort-Noord. Section 8 reports the conclusions and lines of further research.

2. Interlocking description

An interlocking calls and locks safe routes through railway stations and junctions. The interlocking locks a route only if it is safe for a given train, i.e. the route must be vacant from other trains and no conflicting routes must be locked contemporarily. Locking a route for a train means reserving a sequence of track detection sections and locking the direction of movable track elements to allow the train moving from an origin signal to a target signal [25]. Movable track elements include infrastructure elements that have a mutable direction such as switches, level crossings, movable bridges and movable derailment devices [24]. The interlocking typically sets and locks movable track elements in a well-defined direction.

Seven interlocking functions provide safety [25]:

1. Track vacancy acknowledgement by means of track-free detection devices:
   The presence of a train on a track section is typically detected by axle counters or track circuits. Axle counters count the number of train axles that enter and leave a track section. Track circuits detect that train axles short-circuit rails. In Fig. 1, $\overline{T}$ is a joint that electrically isolates two track circuits. In this case, track detection sections t1 and t4 report train occupation to the interlocking and sections t2 and t3 report being vacant. The interlocking will disallow train A to set a route from signal Sig.1 to signal Sig.5 because section t4 is not vacant. A route from signal Sig.1 to Sig.3 can be called if switch Sw.1 is asserted in left direction.
2. Locking the movable track elements of the route:
   If the route is vacant and not claimed by another train, the interlocking commutes the direction of movable track elements. It will lock the direction until the train has released the route. If the signalman requests the route from signal Sig.1 to Sig.3, the interlocking will throw switch Sw.1 to left direction but only if signal 2 is at danger, i.e. shows red and prevents train B from colliding with train A.

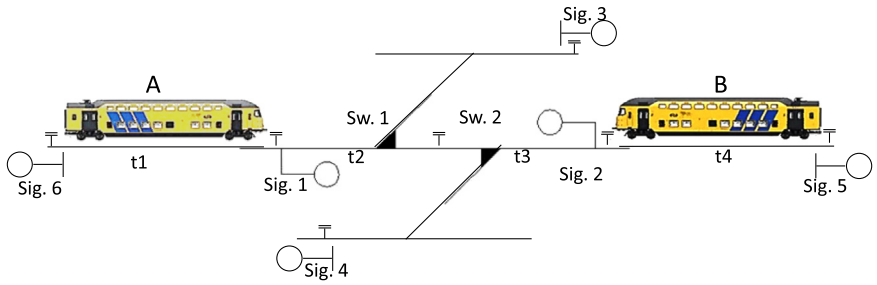


Fig. 1. Simple interlocking area.

3. Route holding:
   After a route is set and movable track elements are locked, this function ensures that the route is held for accommodating the passage of the train. The route is released only after the train has cleared it. Possibly, shorter sections of the route behind the train can be released early and made available for other trains. This increases availability of the network, especially on busy stations.
4. Flank protection:
   Flank protection ensures that trains are not hit by rolling stock from incoming tracks. If a route from signal Sig.1 to Sig.3 is set for train A, signal Sig.3 provides flank protection because train B is forced to stop. As a complementary measure, switch Sw.2 can be locked in left direction such that train B, even if it passes signal 2, cannot hit train A. Whether or not this extra measure is required depends on the level of risk of a flank collision as well as national rules and regulations.
5. Prevent conflicting routes:
   Besides flank protection the interlocking also prevents tail or head-on collision. Signals Sig.1 and Sig.2 are at danger if switches Sw.1 and Sw.2 are in straight direction. In this way a frontal collision is avoided. At this stage, we should point out that the interlocking will only disallow unsafe routes. It will not disallow routes that make no operational sense. The interlocking safeguards from dangerous conflicting routes. It will not prevent conflicting routes that are not dangerous. Human errors can lead to deadlocks such as two opposing trains on a single track with no meet–pass points. Preventing deadlocks is indeed the role of the dispatcher and the timetable.
   The interlocking will at any time ensure that routes are safe and prevent that dispatching and/or driver error induces danger.[2]
6. Overlap protection:
   Trains may overshoot their limit of movement authority, normally red signals, due to poor braking or slippery tracks. The locked route includes a stretch of track beyond the destination signal to allow for overshoot. In Fig. 1, train A may overshoot signal Sig.1. Depending on the length of potential overshoot, the interlocking may have to lock switch Sw.1 or even switch Sw.2. In other words, a prerequisite for setting a route to signal Sig.1 can be that switches Sw.1 and Sw.2 are in a given position. Whether or not this is the case, depends on national rules and regulations, performance of the ATP system and the braking capabilities of rolling stock.
7. Special functions:
   An interlocking can communicate with peripheral systems such as overhead catenary systems or tunnel systems. One may for instance wish to prevent electric powered vehicles entering non-electrified tracks. Alternatively, if a fire is detected in a tunnel, the interlocking automatically forces tunnel entry signals to stop.

This suggests that an interlocking can entertain a relation with external systems, basically any system that can provide binary input (stop/go) that forces chosen signals to stop.

From the main interlocking functions follow the dependencies between signal aspects and directions of movable track elements. Fig. 2 shows the flow of data between an interlocking and other systems that control and monitor traffic. The interlocking controls the aspect of signals, the direction of movable track elements and actuates ATP speed supervision. The interlocking directly communicates with the traffic control system that collects traffic information from the field in the shape of track vacancy detection and signal status. Typically, a signalman calls routes on the non-vital[3] traffic control system which passes the request to the vital interlocking system. The latter checks that all conditions for route setting are met and if so, it sets and locks the route. If the conditions are not met, the signal(s) remain at danger and the traffic control system reports that the route has been refused. Once locked, the interlocking constantly checks that the elements of the route retain the required status. If for instance a track section reports spurious occupation because moisture increases conductivity between rails, the interlocking will instantly ensure that the train stops by forcing the approach signal to show stop.

3. Literature review on interlocking modelling

The railway industry aims at standardizing interfaces and data formats to achieve interoperability among different railway subsystems, simplify the data exchange and the engineering processes [6,22,18,13,14]. In supply chain management the engineering processes are often automated and standardized by integrating the different subsystems of

[2] Certain exceptional situations can require that a signalman or driver overrule the interlocking. Strict regulations, procedures and training are designed to prevent dangerous situations.

[3] Vital systems ensure safety of life whereas command and control systems are classified non-vital.

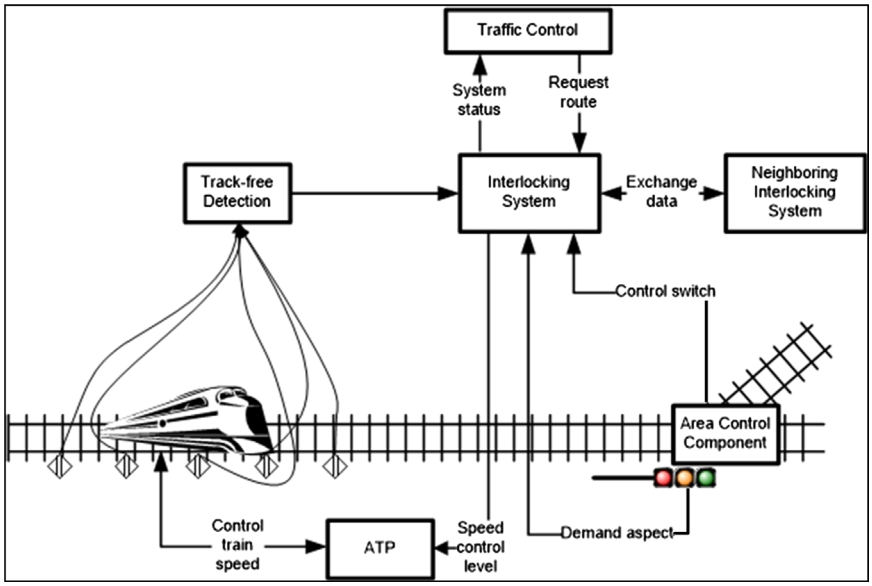


Fig. 2. Communication architecture of a railway interlocking system with peripheral systems for controlling and monitoring traffic.

the chain and using standard machine-readable data for exchanging information [17,23]. Railway infrastructure managers need to automate and standardise the engineering processes of interlocking systems by adopting a standard and machine-readable electronic data format that gives a complete description of the interlocking engineering data. Knowledge-based systems must be developed that can give a standard representation of the information contained in the interlocking system in order to support both designing phases and real-time operations. In the railway sector several knowledge-based systems have been proposed to solve problems relative to the scheduling of train services [8,21], freight cars [11] and for real-time train control [9].

Literature proposes several approaches to describe interlocking systems by means of knowledge-bases (e.g. electronic databases) or also formal models. All of these approaches basically refer to either the "geographical" or the "tabular" family, adopted to represent the interlocking in the practical field. The "geographical" family models the topological structure of the railway network as a graph where nodes are logical objects corresponding to track elements (signals, track detection sections, switches). These objects map to software objects in the memory of the interlocking computer[4]. These objects link according to the track topology. We can consider the signal aspect to be an attribute of a signal object. Also, the aspect of the signal depends on the aspect of the next signal. Thus, we can associate a signal aspect with an edge between successive signal nodes. The edge can have attributes to reflect the signalled ATP speeds. This kind of representation is used by the graphical specification language EURIS (European Railway Interlocking Specification) [4,3,1], used by Siemens. A path algorithm visits the graph node by node. Each node, i.e. switch, section, signal, reports its status, which allows or disallows route propagation. Once all objects on the path report the required status, the route along this path is locked. The geographical interlocking thus implemented the ideas of the IP-protocol way before the internet was conceived.

The "tabular" family represents interlocking dependencies in a table. Such a table collects *per route*, the track objects (switches, track detection sections, signals) plus the state that these objects must take for the route to be allowed. The table reflects the view of the engineer who defines a route as a set of sections that should be vacant, switches that must be in a given direction, etc. Each route is an entry in the table. The number of possible routes through a network increases with the number of switches. This reminds of the early days of internet routing where routes between servers were preconfigured in tables. This approach was feasible for small networks but quickly grew unwieldy as the network grew. Modern Ethernet routers dynamically find routes through the network, irrespective of the number and availability of nodes. It is exactly the same reason why geographical interlocking should be preferred to tabular, for railway networks with larger number of nodes.

As opposed to the geographical approach the tabular interlocking does not compute a route through the network *on the fly*. Geographical interlocking identifies in real-time safe routes by interrogating each node on the path and requesting that that movable object-nodes take a direction that allows the propagation of the path/route. Wang et al. [27] model the interlocking logic by means of an undirected graph representing topological relations among signals and switches. A routing problem is solved by using a shortest path algorithm that searches for feasible routes in a specific direction through the node of this graph by analysing the status of signals and switches (e.g. signal aspect, switch direction) until the correct target

[4] In early geographical interlockings, the objects were assemblies of relays with logic behaviour that implemented the object's methods. In fact, signalling engineers built neural networks *avant la lettre.*

signal of the route is found. Chen et al. [7] propose instead an improved graph representation of the interlocking topology where feasible routes are identified by means of a matrix algorithm that prevents invalid routes based on real-time positions of trains. Roanes-Lozano and Laita [19] provide a topological model of the interlocking logic by means of directed graphs and distinguishing between interlocking areas with and without spring switches. Edges of the graph represent both aspect of signals and direction of switches. Given the requested route and the current positions of trains, conflicting routes are identified by means of Boolean matrices. Later, the same authors [20] introduced a model of the interlocking logic that is independent from the topology of the network and is based on the use of polynomial ideals and Gröbner bases for a more concise description of the interlocking relations.

Banci [1] describes the interlocking logic by means of state charts, assuming that signals, tracks and movable track elements are machines with a corresponding state (e.g. the aspect for signals and the direction of switches). Dependencies between signals and switches are therefore modelled as a dynamic system where the state transition of a machine, e.g. the aspect of a signal, depends on the state of other machines of the system, e.g. on the direction of switches.

Models based on the tabular approach rely on a predefined set of routes. When calling one of these routes, conflicting routes will be blocked. Several models have been proposed based on the concept of the tabular representation. Among these it is worth mentioning the approach of Lehmann and Albrecht [13] who developed a base of knowledge that collects the characteristics of all the physical components of the interlocking system, and use the RailML markup language to standardise the data format. Infrastructure elements, e.g. signals, signs, and track detection sections, are used to delimit the start and the end of possible routes which are identified in terms of the corresponding setup/release times, the sequence of track sections composing them, flank protection elements and overlap rules. Fries [10] proposes collecting interlocking characteristics by combining the advantages of typical electronic databases (e.g. MS Access) with those of standard XML to represent railway data, i.e. RailML. Having a database with relations to the RailML scheme, overcomes the limits of databases that cannot capture all the characteristics of track elements and the limits of RailML. These often do not contain data at the level of detail necessary for describing the interlocking logic. In this case a tool can convert information from the database to RailML and vice versa. The RailML group [16] initiated a RailML schema for describing interlocking systems, including feasible routes, and switch-signal dependencies.

Summarising, the geographical model contains the interlocking topology and the aspects of signals but does not define routes, since routes are dynamically determined by algorithms. The tabular model lists a predefined set of possible routes but is unaware of topology.

4. The railway markup language: RailML

The Railway Markup Language RailML is an open source XML-based description language started in 2002 to standardise data exchange between different IT railway systems. Hürlimann and Krauss [12] and Nash et al. [15] introduced this XML-based language, showing the flexibility and the effectiveness of its structure for collecting and exchanging railway data. RailML is being increasingly used, in Europe and beyond, by railway industry and academia.

Three RailML schemes have been defined to capture infrastructure, timetable and rolling stock. A fourth scheme for capturing interlocking data is under development. Linder and Grimm [14] illustrate that the RailML language is also suitable to represent the characteristics of interlocking systems.

A RailML file that describes the infrastructure, contains all the information about the tracks such as: their length, gradient, speed limit, radius; position of switches, signals, track vacancy detectors, platforms, etc. A RailML timetable file holds information concerning the train services (i.e. scheduled arrival/departure times) including scheduled routes. The rolling stock scheme describes all the physical, mechanical and electric characteristics of the rolling stock, e.g. mass, resistance factors, traction, braking curves, length, number of coaches.

An entry in a RailML timetable file, i.e. a train service, will typically refer to a trainset, defined in terms of the rolling stock scheme, and a scheduled route defined in terms of the RailML infrastructure.

The RailML standard requires that all elements have a unique identification number *id* to ensure non-ambiguous referral between elements.

RailML is best explained by an example. We chose a RailML representation of a minor piece of infrastructure because infrastructure is obviously relevant to interlocking whereas timetables and rolling stock are not. Fig. 3 shows the UML diagram of RailML version 2.2. Element “tracks” is a child of root element “Infrastructure” and contains all the tracks of the network. The railway network is modelled as a graph where a track is a directed arc between two nodes named “trackBegin” and “trackEnd”. These two nodes belong to the element “trackTopology” which also contains “connections” to switches or other tracks. Fig. 4 shows how RailML models tracks $T_{i=1..4}$ as directed edges delimited by nodes $N_{j=1..5}$. The RailML snippet describes the topology of track T_2.

This snippet shows that the beginning node of track T_2 is N_2 located at position 15 while the end node is N_3 at position 35. These nodes connect respectively with track T_1 and T_3 by means of the connections “$N_{2,T1}$–$N_{2,T2}$” and “$N_{3,T2}$–$N_{3,T3}$”. Zooming in on node N_3 we explain how RailML models connections among tracks. $N_{3,T2}$ is the edge of arc T_2 converging in node N_3. $N_{3,T3}$ is the edge of arc T_3 diverging from node N_3; while $N_{3,T4}$ is the edge of arc T_4 converging in node N_3. RailML specifies that node N_3 connects edge $N_{3,T2}$ of arc T_2 to edge $N_{3,T3}$ of arc T_3. Coupling the names of these two edges, gives the identifier *id* of such a connection: $N_{3,T2}$–$N_{3,T3}$. The *ref* attribute is $N_{3,T3}$–$N_{3,T2}$ which coincides with the “id” of the same connection seen from the perspective of arc T_3. Same rules hold for the connection with arc T_4 which is a switch. In this case beside the attributes *id* and *ref*, the RailML specifies the *course* (i.e. the direction of the switch: left, right or straight) and the orientation (i.e. incoming or outgoing) with respect to the direction of arc T_2.

By following the direction of arc T_2 switch T_4 is located to the right side, that is why the *course* is right. The orientation is instead incoming into arc T_2.

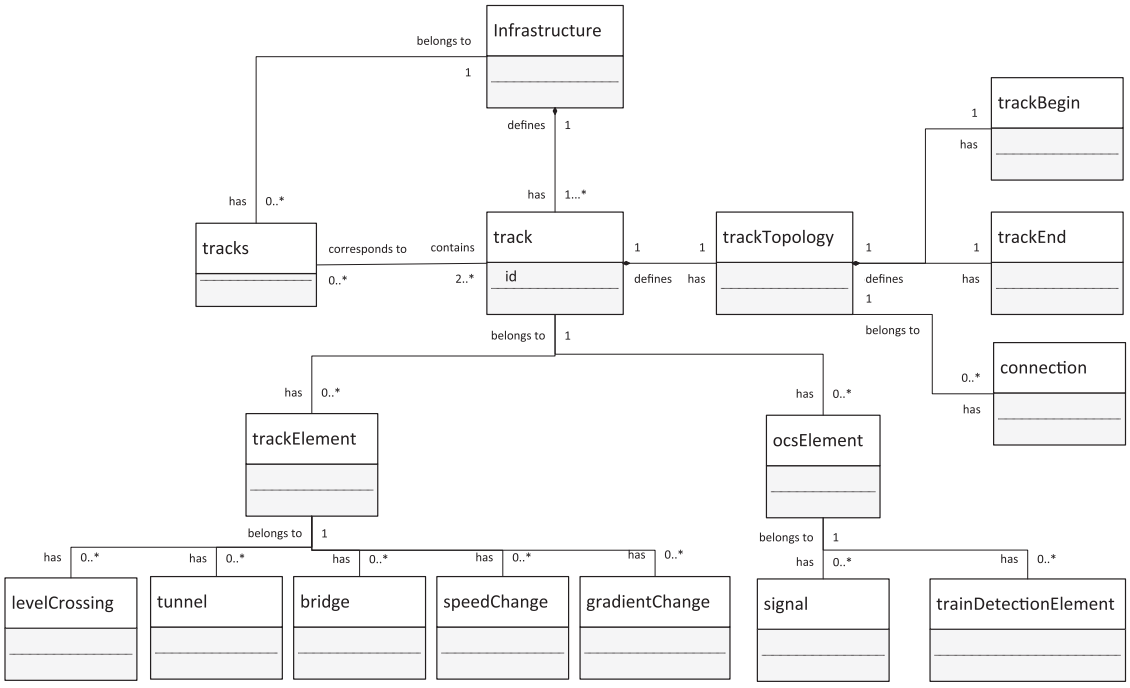
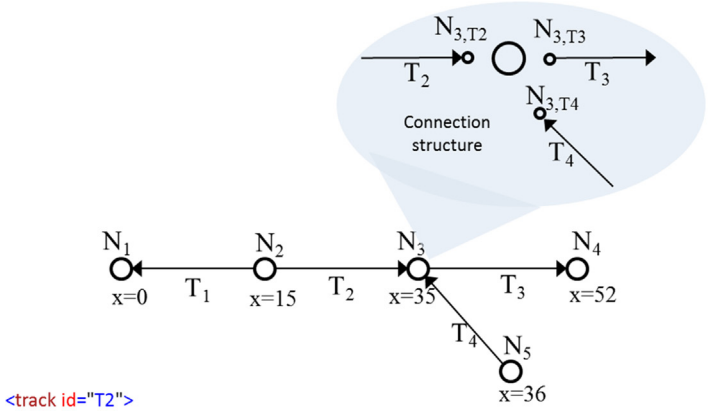


Fig. 3. Scheme for representing the infrastructure in RailML version 2.2.




```
<track id="T2">
  <trackTopology>
    <trackBegin id="N2" pos="15">
      <connection id="N2,T2-N2,T1" ref="N2,T1-N2,T2"/>
    </trackBegin>
    <trackEnd id="N3" pos="35">
      <connection id="N3,T2-N3,T3" ref="N3,T3-N3,T2"/>
    </trackEnd>
      <connections>
        <switch pos="35" id="T4">
          <connection course="right" orientation="incoming" id="N3,T2-N3,T4" ref="N3,T4-N3,T2"/>
        </switch>
      </connections>
  </trackTopology>
</track>
```

Fig. 4. Topological representation of infrastructure tracks in RailML.

Beside topology, Fig. 3 shows other elements, in particular child node “trackElement” carrying physical information of the track such as the gradient, the radius, the speed limit, the presence of tunnels, bridges and/or level crossings. Child node “ocsElements” contains operation and control system elements, i.e. type and position of signals and track-vacancy detectors (c.f. [16]).

The RailML infrastructure model describes network topology and signals in a format that allows for easy extraction of routes, signals and switches that one encounters en route. This information is essential to interlocking engineering.

5. Current approaches used in practice to represent interlocking data

Section 3 points out that an interlocking can have either a tabular or a geographical representation. The tabular interlocking is unaware of network topology and collects interlocking dependencies for a set of predefined routes.

The geographical interlocking sees the railway network as a graph where routes are paths through the graph. As opposed to the tabular representation, the geographical interlocking is aware of network topology and it has no need to predefine routes since they are dynamically identified by means of algorithms.

5.1. The tabular interlocking perspective

The tabular interlocking collects into a table the interlocking dependencies between signals and direction of movable track elements, needed for a safe route. The interlocking fed a predefined set of *allowed* routes which is a subset of all routes that are physically *possible*. Usually, the IM provides a list of allowed routes that make operational sense. Each route is associated with a set of signals and switches that must be in a given state. A route is locked only when these signals and switches are in the required state. As an example, Table 1 provides the tabular representation of the interlocking area depicted in Fig. 5.

Table 1
Tabular representation of the interlocking area in Fig. 5.

Route	Signal A	Switch 1	Switch 2	Switch 3	Switch 4	Signal C
A→B 1	Green	S	S	S	D	Red
A→B 2	Green	D	S	S	S	Red
A→B 3	Green	D	D	S	D	Red

Specifically this table lists the dependencies between signal aspects and switch directions to set three routes from start signal A to target signal B. The tabular representation does not directly convey information about the topology of the interlocking area. Instead for every predefined route (i.e. Routes 1, 2, and 3) the table lists the aspect of each signal A, B and C and the direction of switches 1, 2, 3, and 4 that make it a safe route. Aspects are typically Green or Red, and the direction of switches are typically straight (S) or diverging (D). Note that the state of signals and switches also provides flank protection. Route 3, whilst perfectly safe, is operationally unusual and the IM may choose not to implement this route in his interlocking.

These routing tables are hard to maintain when nodes (i.e. signals and switches) are added or removed from the interlocking.

The internet protocol introduces the concept of dynamic routing. Each node in the network is connected to adjacent nodes and there is no need to predefine routes, as the protocols reroute packets as nodes are added or removed. Similarly, routing tables for tabular interlocking grow exponentially with the size and complexity of the station. Therefore, in these cases it is better to switch to a geographical representation which finds safe routes dynamically and does not need them to be predefined.

5.2. The geographical perspective

A geographical interlocking is aware of the network topology and unaware of routes. Indeed this approach models the interlocking area as a graph where nodes represent signals and movable track elements, while arcs depict track elements connecting nodes. Fig. 6 gives the geographical representation of the interlocking area illustrated in Fig. 5. In particular we refer to the geographical

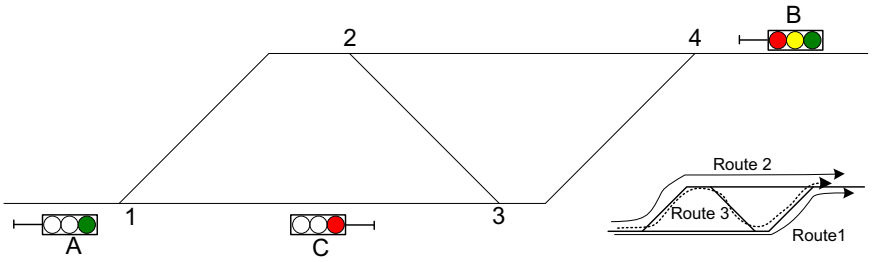


Fig. 5. Schematic representation of an interlocking area.

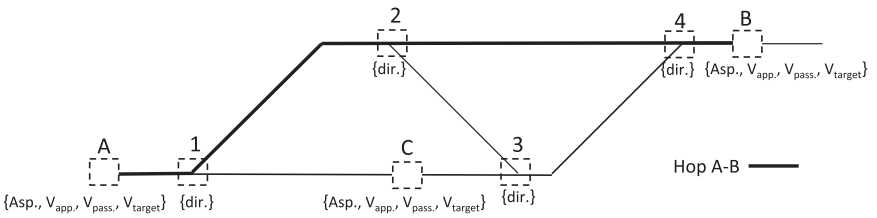


Fig. 6. Geographical representation of the interlocking area in Fig. 5.

representation adopted by the specification language EURIS [4] which depicts signals and switches as objects with a set of attributes and rules. Nodes depicting signals have four attributes: the aspect (Asp), the speed limit when passing the signal (V_{pass}), the speed limits when passing the previous ($V_{approach}$) and the next signal (V_{target}). The three speed limits are imposed by line-side speed boards, national rules and/or the ATP system. It should be noted that the speed attributes are specific to signalling systems that transmit maximum speed to trains. Other signalling systems may associate signals with attributes that indicate for instance direction or destination.

Nodes representing switches have a direction attribute (dir). The direction is either straight or diverging though one can also define a preferred direction. If a train wishes to travel between two signals, a path search algorithm visits the graph node by node and identifies the route considering the current aspect of signals and the direction of switches. The interlocking is aware of the preferred switch direction which is diverging for switch 1 and straight for switch 4. Given this preference, a shortest path algorithm such as Dijkstra's can determine a unique route between two consecutive signals A and B. This approach requires no predefined route tables as in the tabular approach. As a consequence, modification of the network is easy. Similar to the internet protocol, the route-finding algorithm is immune to removal or adding of nodes from the network. For this reason the geographical interlocking is usually preferred for large and complex railway stations.

Again following the computer networking analogy, we can reason in terms of *hops* where a hop conveys a packet from node to node. In this case a hop coincides with the shortest route from a start signal to a target signal. An example of a hop from A to B is highlighted with a black bold line in Fig. 6. A train moving between two signals will therefore hop from one signal to the next. A hop, hence the shortest safe possible route between a start and a target signal has a number of attributes:

- Aspect of start signal (Red, Yellow, Green, etc.).
- Aspect of target signal.
- Approach speed for start and target signals.
- Passing speed for start and target signals.
- Target speed for start and target signals.
- Direction of switches encountered en route.

Specifying these attributes completely defines a route between two signals. This leads us to a model that focuses on signals and hops (or relations) between signals.

6. The proposed interlocking data representation

Our aim is to reduce the inefficiency, time, and cost of the engineering processes of interlocking systems by developing a machine-readable and standardized base of knowledge of the interlocking. A similar database can strongly improve the engineering processes because: i) stakeholders do not need to collect interlocking data from the field, since a single electronic database contains all information; ii) automatic data conversion means lower probability of human interpretation error; iii) there is easier communication between parties since the same base of knowledge reduces misunderstandings [5]; and iv) fewer conversion errors because a standard formal data exchange language means economy of scale with respect to the present situation where signalling industry has to convert data from heterogeneous non-standard national formats.

A base of knowledge that fully maps the engineering of the interlocking system must describe not only the interlocking topology but also the signal aspects, the

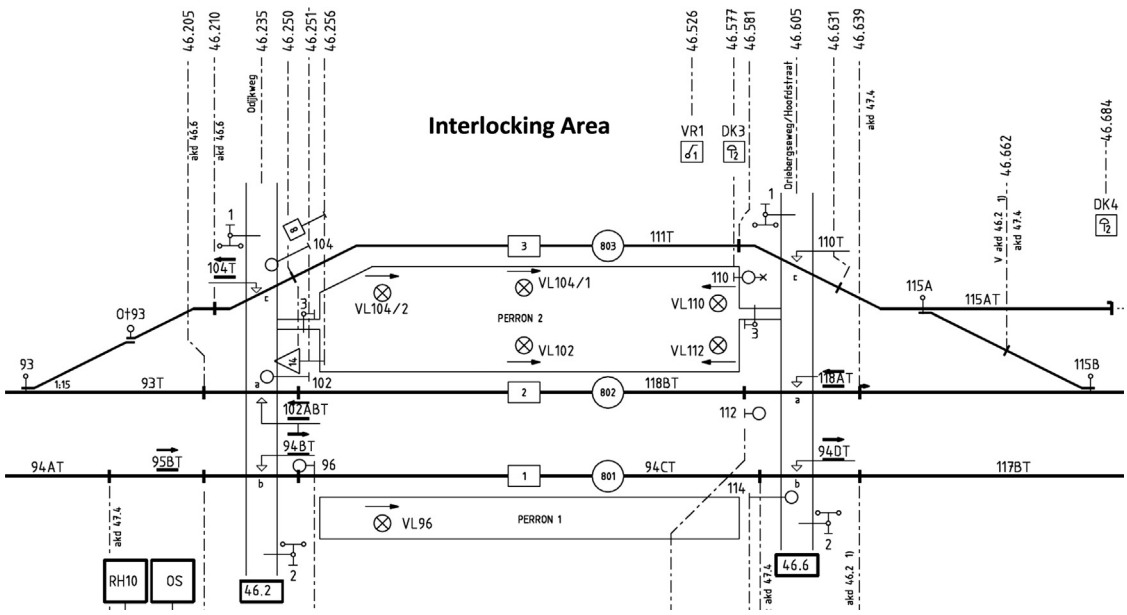


Fig. 7. Track plan showing the topology of an interlocking area.

speed indications from line-side signs and/or the ATP system, as well as the dependencies between switches and signals as well as the set of possible safe routes. Section 5 suggests that a base of knowledge based only on the geographical interlocking would not be enough, since the static information on dependencies between switches and signals is missing. On the other hand, a base of knowledge adapted to the tabular interlocking would not suffice either, because it does not provide the topology that a complete interlocking representation needs.

For this reason we develop a base of knowledge that couples the geographical and the tabular interlocking representations. Topology, signalling and ATP rules, routes and the dependencies between signals and switches are collected from technical documents and stored in a format that is proposed for RailML interlocking scheme. The main advantage of our approach is that the interlocking database can be easily created for whatever interlocking, just starting from technical documents usually used in practice.

6.1. Topology

Our database describes network topology by means of the RailML infrastructure scheme that we explained in Section 4. Main inputs to this scheme are the track plans, an example of which is shown in Fig. 7. Apart from topology, the track plan typically provides (co-)ordinates of infrastructure elements such as track circuits, switches, and signals.

Most information is captured by the RailML infrastructure model where switches, and buffer stops are nodes while tracks between nodes are arcs.

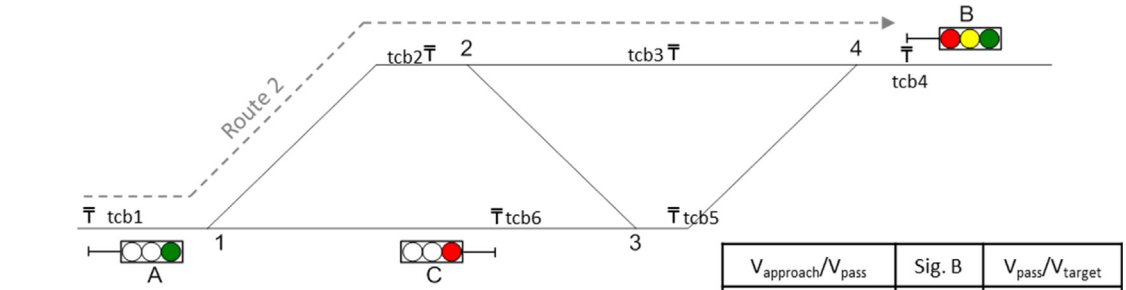
The RailML infrastructure scheme captures physical connections between infrastructure elements but does not consider functional connections, such as interlocking dependencies between signals and switches, or flank protection. This is one lack in the infrastructure scheme that our approach overcomes by extending the RailML standard with an interlocking scheme that efficiently describes these functional connections. The proposed RailML database of the interlocking must therefore be used always together with the RailML infrastructure scheme. This latter provides indeed the topological characteristics of all the elements contained in the interlocking database.

6.2. Dependencies between signal aspects and speed indications

When a route is set, the start and the target signals assume specific aspects which allow one specific train to safely travel that route while keeping out other trains. Often, speed indications are associated with signal aspects. Signal plans graphically represent the possible sequence of signal aspects that a train may encounter.

In other words, signal plans describe the relations between aspects of signals. These drawings are highly readable by humans but not by computers. Engineers must extract speed information from signal plans that the interlocking needs when actuating ATP which safeguards against overspeed and signals passed at danger.

Fig. 8 illustrates a signal plan of the interlocking area in Fig. 5. This plan refers to the Dutch signalling system, although the general idea applies to any other speed signalling system. The Dutch signalling has few aspects; Green (G), Yellow (Y) and Red (R). Additional aspects like Green Flashing (G_{FL}) and Yellow Flashing (Y_{FL}) warn the driver to adopt a cautious driver behaviour as specified in [25]. We restrict the sample to signals A, B and route 2. Symbols $\overline{\top}$ are track circuit joints. Each signal has an "aspect-speed" table that lists the set of possible aspects and the corresponding ATP speed limits $V_{approach}$, V_{pass} and V_{target} in Km/h. Links between the rows of the tables represent the dependencies among signal aspects of two



$V_{approach}/V_{pass}$	Sig. A	V_{pass}/V_{target}
130/0	R	
130/40	Y	40/0
130/40	G_{FL}	40/40
130/130	Y	130/0
130/130	G	130/130

$V_{approach}/V_{pass}$	Sig. B	V_{pass}/V_{target}
40/0	R	
40/0	Y_{FL}	
40/40	Y	40/0
40/40	G_{FL}	40/40

Fig. 8. Simplified Dutch Signal plan.

consecutive signals. Once the aspect of a signal is known we identify the aspect of the other signal by simply following the links. We read the signal plan as follows:

If signal A shows Green-Flashing (G_{FL}), the train may approach signal A at $V_{approach}=130$ km/h. But it must have slowed down to $V_{pass}=40$ km/h at signal A, and remain at $V_{target}=40$ km/h to target signal B. Signal B can show either Yellow or Green-Flashing.

If signal A is yellow the train can pass the previous signal at $V_{approach}=130$ km/h, slow down to $V_{pass}=40$ km/h at signal A and stop ($V_{target}=0$ km/h) at the target signal B. Signal B has a Red aspect or Yellow Flashing if the distance between signals A and B is not enough to brake the train from 40 km/h to 0. In this case the train must stop at the signal following signal B.

Our objective is to formally express these dependencies. We use object-oriented modelling [2,3] and represent each signal as an object characterised by a set of attributes. Every signal is identified by a unique identifier *ref* pointing to a matching *id* in the infrastructure schema from which we may easily retrieve its topological characteristics. Each row of the aspect-speed table of a signal constitutes a dependency between a signal aspect and an ATP speed indication that is associated with that signal. Each dependency, i.e. each row of the table, can be modelled as a tuple $(aspect, V_{approach}, V_{pass}, V_{target})$. Therefore the "aspect-speed" table associated with a signal is a set of aspect-speed tuples. For instance the aspect-speed table of signal A is represented as follows:

$$\text{Aspect} - \text{speed dependencies of signal A} = \begin{pmatrix} (\text{R}, 130, 0, 0) \\ (\text{Y}, 130, 40, 0) \\ (\text{G}_{FL}, 130, 40, 40) \\ (\text{Y}, 130, 130, 0) \\ (\text{G}, 130, 130, 130) \end{pmatrix}. \quad (1)$$

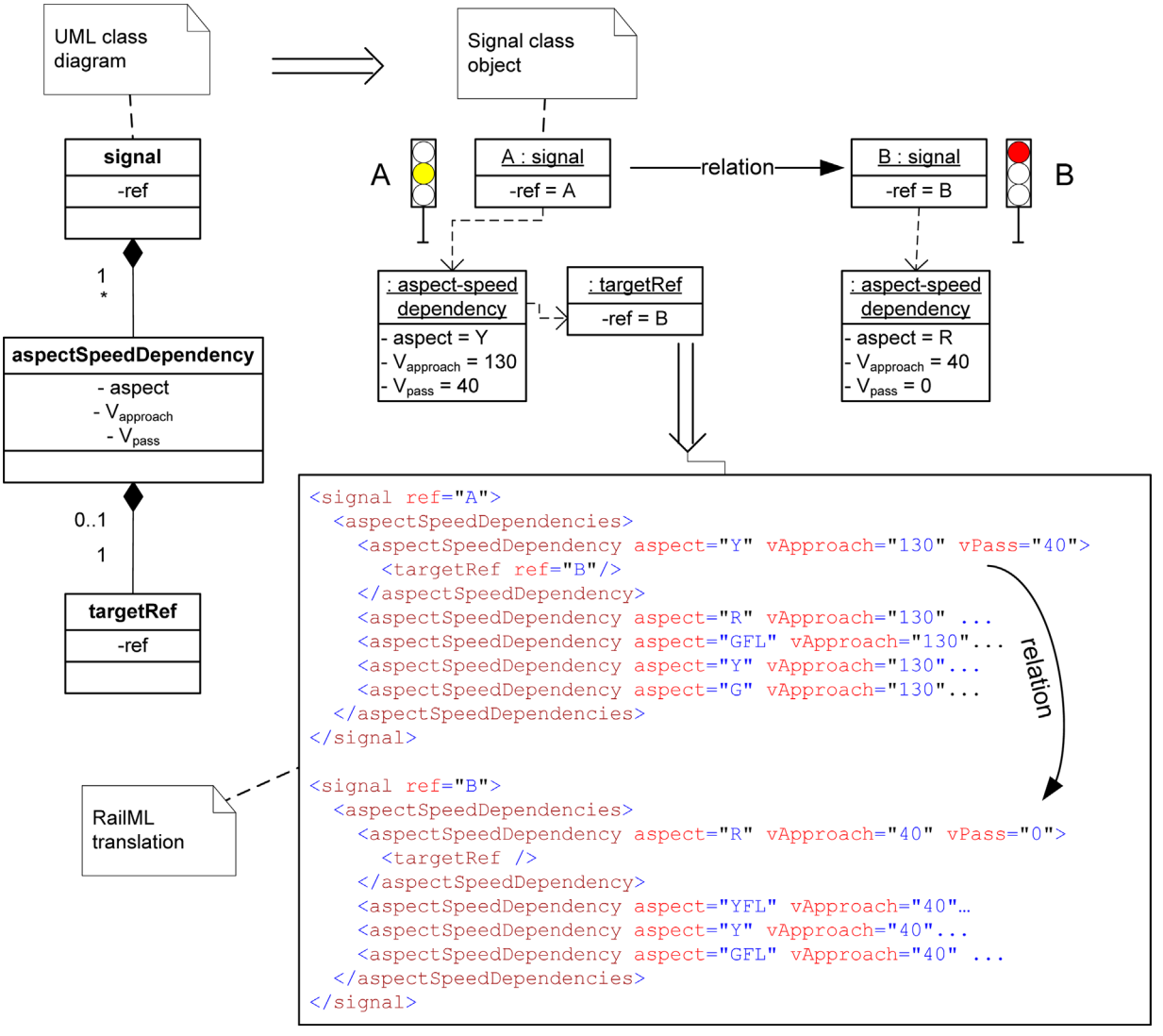


Fig. 9. UML class diagram, class object representation and RailML translation of signals with their aspect-speed dependencies.

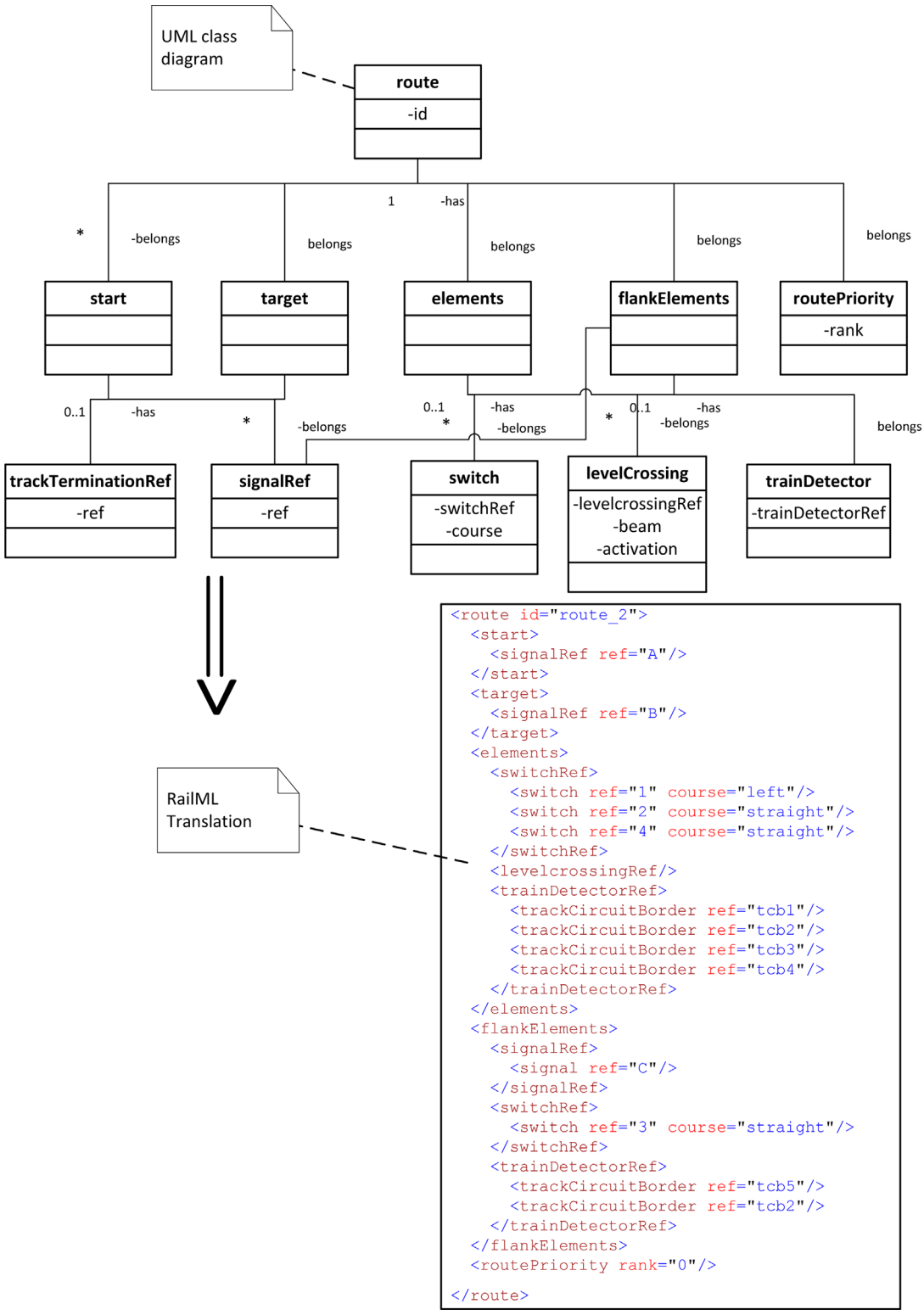


Fig. 10. UML class diagram and corresponding RailML translation used to collect data of routes and relations.

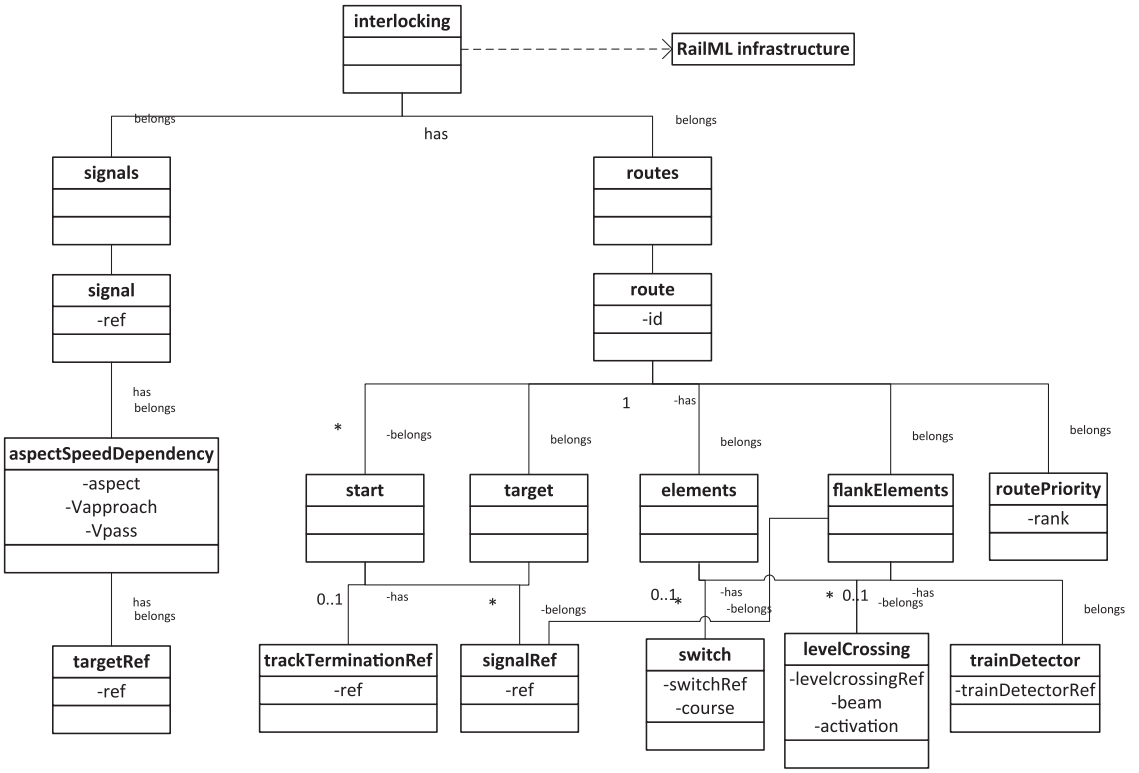


Fig. 11. UML class diagram at the basis of the proposed RailML interlocking database.

At this point the links between the aspect-speed tables of two consecutive signals can be modelled as relations between tuples. The link between an aspect of the start signal i and an aspect of target signal j can be mathematically expressed as:

$$(aspect, V_{approach}, V_{pass}, V_{target})^{i} \rightarrow (aspect, V_{approach}, V_{pass}, V_{target})^{j} \quad \forall (i,j) \in S \tag{2}$$

where S represents the set of all adjacent signals in the interlocking area.

From the relation reported above it is easy to understand that the target speed V^{i}_{target} of the i-th start signal coincides with the passing speed V^{j}_{pass} of the j-th target signal, i.e. the condition $V^{i}_{target} = V^{j}_{pass}$ holds. This equality is true because of the definition of V_{target} which is indeed the passing speed V_{pass} at the target signal. The target speed is therefore a redundant element in the relation (2) and can be left out. This means that in our model we will refer not anymore to the tuples of four elements $(aspect, V_{approach}, V_{pass}, V_{target})$ but to tuples of three elements $(aspect, V_{approach}, V_{pass})$ that do not consider the redundant element V_{target}. In this case the relation (2) becomes:

$$(aspect, V_{approach}, V_{pass})^{i} \rightarrow (\text{aspect}, V_{approach}, V_{pass})^{j} \quad \forall (i,j) \in S. \tag{3}$$

In a database, the relation expressed in (3) must be captured by means of an attribute of the tuples in order to keep track of the relation without introducing any redundancy nor infringing data integrity. A suitable attribute for this purpose is the identifier *ref* of the target signal which the relation refers to. If *targetId* is the unique identifier of the j-th target signal, we can express relation (3) as:

$$(aspect, V_{approach}, V_{pass})^{i}_{targetId} \quad \forall\, i \in S \tag{4}$$

In this way the relations with the aspects of the j-th target signal are expressed by means of an attribute of tuples of the i-th start signal.

According to this modelling approach the set of aspect-speed dependencies (i.e. the aspect-speed table) of signal A reported in (1) translates into:

$$\text{Aspect} - \text{speed dependencies of signal A} = \begin{pmatrix} (\text{R}, 130, 0) \\ (\text{Y}, 130, 40)_{B} \\ (\text{G}_{FL}, 130, 40)_{B} \\ (\text{Y}, 130, 130) \\ (\text{G}, 130, 130) \end{pmatrix} \tag{5}$$

where the second and third tuples have the identifier of target signal B as an attribute to indicate their relations with this signal.

When converting the above notions into a UML diagram we obtain the scheme illustrated in Fig. 9 that provides the structure of the object class “signal” on which we build our database.

Signals A and B in the interlocking area of Fig. 8 are objects with attributes and relations. For the sake of simplicity we only report one of the several aspect-speed dependencies of both signals. The translation of these objects into RailML produces the excerpt of our RailML

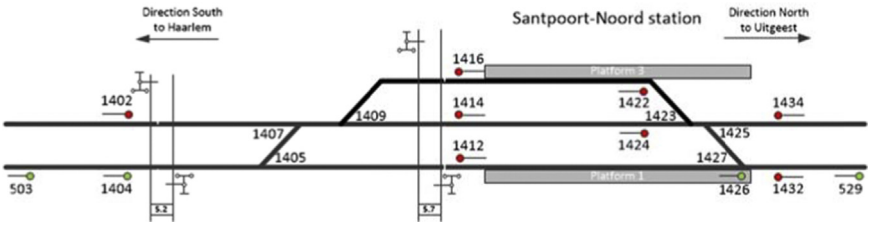


Fig. 12. The interlocking area of Santpoort Noord.

database where the arrow visualises a relation between Signal A, Yellow and Signal B, Red.

Aspect-speed dependencies are children of a *signal* object and have aspect, $V_{approach}$and V_{pass} as attributes. These triplets keep track of the relations between aspects of consecutive signals by referring to the target signal by means of the attribute *targetRef*.

Our model is a lean translation of the signal plan and a contribution to the state of the art since it overcomes one of the main limits of the database formulations previously proposed in literature (e.g. [13,14]).

6.3. Dependencies between track elements within routes

Apart from implementing the signal plan, the interlocking must assert that a well defined set of track elements are in a given state when calling a route. This means that track detection sections must be vacant, switches in a given position, and level crossings closed.

The required element status is mostly taken from track and signal plans. Track plans or design documents often provide information about flank protection that is reflected by required status of signals and switches that are off-route.

In addition, each route must have a priority to distinguish the default route from other alternative routes that can only be set if specifically requested by the signalman. The interlocking will automatically set the default route if not differently established by the signalman. To this purpose a priority rank is assigned to routes. The lower the rank, the higher is the priority of the route. The default route has therefore the lowest priority rank. Fig. 5 shows that three routes exist between signals A and B. If route 2 is the default route a rank=0 is attributed to it, while routes 1 and 3 will have respectively rank 1 and 2.

Fig. 10 shows the UML class diagram designed to collect information on routes and track element dependencies. Again we rely on object-oriented modelling by defining the route as an object with the identifier *id* as an attribute. The route has five children which are: the start and target objects, the elements composing the route, the elements providing flank protection and the priority of the route. The start and target objects of a route generally coincide with signals but they can also correspond to track terminations. For this reason, both start and target of the route have two children *signalRef* and *trackTerminationRef* with attribute *ref* pointing to either a signal or a track termination in the RailML infrastructure scheme. The class *elements* identifies all those en-route elements composing the route, while the class *flankElements* represents all those off-route elements providing flank protection. Children of these two classes are switches, level crossings and train detection devices. Switches have attributes *switchRef* and *course*. The former attribute refers to the switch in the RailML infrastructure scheme, while the latter gives the direction of the switch. Level crossings have as attributes the identifier of the level crossing (*levelCrossingRef*), the identifier of the device activating the level crossing (*activation*), as well as the direction that the *beam* must have to safely set the route. The class representing train detectors has only attribute *trainDetectorRef* which is the identifier of the train detection device as defined in the RailML infrastructure scheme. The class *flankElements* has the additional child *signalRef* which refers to a signal providing flank protection. When a signal provides flank protection, it must be always restrictive if the parent route is called.

The class *routePriority* defines the priority *rank* of routes, if more route alternatives are available between the same start and target objects.

The (proposed) RailML code in Fig. 10 models route 2 of Fig. 8 including relations between track elements. It states the start and target signals of the route, the switches including their direction, the borders of the track circuits en route, and the elements providing flank protection. Flank protection is given by signal C showing red, switch 3 in straight direction and the track circuit between *tcb5* and *tcb2* being vacant.

6.4. UML structure of the proposed interlocking data representation

The RailML interlocking database proposed in this paper is based on the UML class diagram shown in Fig. 11 which is obtained by combining the UML schemes defined for the classes “signal” and “route”, respectively. This database contains therefore the dependencies among all the elements of the interlocking system: signals, movable and non-movable track elements. When this database is linked (see dotted arrow) with the RailML infrastructure that describes the topology, the interlocking system is completely described from both the physical and behavioural points of view. As can be seen the class *interlocking* is composed of the two children elements *signals* and *routes*. The former collects all the signals of the interlocking area which are modelled as described in Section 6.2 to catch all their aspect-speed dependencies and their relations with the other signals. The latter on the other hand lists the possible routes between two consecutive points (either signals or track terminations) describing them as illustrated in Section 6.3. In the following this UML scheme

```
<interlocking>
 <signals>
  <signal ref="Sptn_1404">
   <aspectSpeedDependencies>
    <aspectSpeedDependency aspect="R" vApproach="130" vPass="0">
     <targetRef ref="Sptn_1422"/>
     <targetRef ref="Sptn_1424"/>
     <targetRef ref="Sptn_1426"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YLFL" vApproach="130" vPass="0">
     <targetRef ref="Sptn_1422"/>
     <targetRef ref="Sptn_1424"/>
     <targetRef ref="Sptn_1424"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YL" vApproach="130" vPass="40">
     <targetRef ref="Sptn_1422"/>
     <targetRef ref="Sptn_1424"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YL" vApproach="130" vPass="80">
     <targetRef ref="Sptn_1426"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="GRFL" vApproach="130" vPass="40">
     <targetRef ref="Sptn_1422"/>
     <targetRef ref="Sptn_1424"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="GR" vApproach="130" vPass="130">
     <targetRef ref="Sptn_1426"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YL" vApproach="130" vPass="130">
     <targetRef ref="Sptn_1426"/>
    </aspectSpeedDependency>
   </aspectSpeedDependencies>
  </signal>
  <signal ref="Sptn_1426">
   <aspectSpeedDependencies>
    <aspectSpeedDependency aspect="R" vApproach="80" vPass="0">
     <targetRef ref="Bv_529"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YLFL" vApproach="80" vPass="0">
     <targetRef ref="Bv_529"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YL" vApproach="130" vPass="80">
     <targetRef ref="Bv_529"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="GR" vApproach="130" vPass="130">
     <targetRef ref="Bv_529"/>
    </aspectSpeedDependency>
    <aspectSpeedDependency aspect="YL" vApproach="130" vPass="130">
     <targetRef ref="Bv_529"/>
    </aspectSpeedDependency>
   </aspectSpeedDependencies>
  </signal>
 </signals>
 <routes>
  <route id="Sptn_1404_1426">
   <start>
    <signalRef ref="Sptn_1404"/>
   </start>
   <target>
    <signalRef ref="Sptn_1426"/>
   </target>
   <elements>
    <switchRef>
     <switch ref="Sptn_1405" course="straight"/>
    </switchRef>
    <levelcrossingRef>
     <levelcrossing ref="Hlm_137bV_Sptn_1405V_5.290" beam="down"/>
     <levelcrossing ref="Sptn_1405R_1427L_5.76" beam="down"/>
    </levelcrossingRef>
    <trainDetectorRef>
     <trackCircuitBorder ref="Sptn_1405V_100"/>
     <trackCircuitBorder ref="Sptn_1405V_300"/>
     <trackCircuitBorder ref="Sptn_1405V_600"/>
     <trackCircuitBorder ref="Sptn_1405R_200"/>
     <trackCircuitBorder ref="Sptn_1405R_400"/>
     <trackCircuitBorder ref="Sptn_1405R_600"/>
    </trainDetectorRef>
   </elements>
   <flankElements/>
<routePriority rank="0"/>
  </route>
 </routes>
</interlocking>
```

will be applied to create the RailML database of a real interlocking area, to show the effectiveness and the completeness of the approach.

7. Application to a real case study

We evaluate our proposed approach by applying it to the Santpoort Noord station in the Netherlands, illustrated in Fig. 12. In spite of its small size, there are 12 signals, six switches and two level crossings.

The topological data of this interlocking area are picked from a track plan and converted into RailML. The signal plan provides interlocking relations between signal aspects, switch directions and ATP speed indications. Below we report an excerpt of our interlocking database for signals 1404 and 1424 and the only possible default route between them:

This RailML database matches the UML class diagram in Fig. 11. The RailML database of the whole station can be downloaded together with the XML Schema Definition (XSD) and additional documentation from the TU Delft data archive [26]. At this point we compare our approach with the tabular and the geographical interlocking representations. As stated before, the data structure for tabular interlocking does not directly provide information about topology nor relations between aspect and speed indications for signals. On the other hand the data structure of a geographical interlocking does not convey static information about routes between signals. The comparison is both qualitative and quantitative and the outcomes are shown in Table 2. In particular the qualitative criteria refer to the capability of modelling dependencies among track elements en route and those used for flank protection.We also look at the possibility of representing signal aspects and their relations with ATP speed indications as well as

Table 2
Qualitative and quantitative comparison between the discussed formalization methods.

Criteria	Data structure at the basis of the database		
	Proposed interlocking representation	Tabular interlocking	Geographical interlocking
Qualitative			
Dependencies of track elements en route	Yes	Yes	No
Dependencies of track elements on flank	Yes	Yes	No
Signal aspects	Yes	Yes	No
Dependencies of signal aspects	Yes	Yes	Yes
Dependencies signal aspect-ATP speed indications	Yes	No	Yes
Route priorities	Yes	Yes	No
Quantitative			
Amount of lines	128	196	65
File size	4183 bytes	6146 bytes	1666 bytes

with the aspect of the successor or predecessor signals. We then compare the size (in bytes and number of lines) of the resulting databases.

As opposed to the strictly tabular and the geographical interlocking, our representation captures all the interlocking dependencies, suggesting complete coverage. Our model is of course larger than the geographical one, since this latter does not model routes. Our approach is much more concise than the tabular interlocking since our signal representation removes redundancy in route elements.

8. Conclusion

The design and engineering of railway interlocking systems are too expensive due to the complex and inefficient iterative exchange of technical information between stakeholders. To reduce waste of time and money, the railway sector wishes to reduce complexity by introducing standards for the exchange of complete and error-free information.

Errors due to human misinterpretation and data-conversion from non-standard formats are the main cause of waste that must be mitigated by costly iterations of corrections and tests.

Most references in literature describe either tabular or geographical interlocking. Either representation is incomplete. The tabular representation fails to represent the network's topology while the geographical fails to represent routes. We propose a model that combines the best of the tabular and the geographical approaches. We defined an UML class diagram to collect all interlocking dependencies. It can capture the topology of the interlocking as well as all possible routes, the relations between signal aspects, ATP speed indications and the state of both movable and non-movable track elements. This model is a proposition for the RailML standard which, as an incarnation of XML, is by definition machine-readable. The defined UML class diagram has been applied to depict the interlocking area Santpoort-Noord in the Netherlands. The resulting RailML database is complete and concise.

As opposed to other approaches, the interlocking model based on our UML class diagram completely maps every feature of the interlocking and as concisely as possible. Demonstrating this utility is the main contribution to the scientific literature. The fact that this model arose in a joint collaboration with Dutch railway infrastructure manager ProRail and the signalling supplier Siemens underwrites the real need for such a model. Many other European infrastructure managers are interested to the proposed approach for the reasons set out above.

The applicability of our approach is general since the proposed database is easily built by taking data from infrastructure managers and technical documents, such as track plans or signal plans. Our approach can fit all signalling systems, ATP, interlocking systems, rules and regulations.

Future research is planned to investigate automated creation of RailML from existing databases, modelling other signalling systems and ATP systems, and in particular integration with ETCS.

Acknowledgements

The authors thank the Dutch railway infrastructure manager ProRail for the support to this research and the provision of all the necessary interlocking data needed for testing the proposed approach. A particular thank you goes to the reviewers who helped the authors to improve the structure and legibility of the paper.

References

[1] M. Banci, Geographical versus functional modelling by statecharts of Interlocking Systems, Electron. Notes Theor. Comput. Sci. 133 (2005) 3–19.
[2] J. Brunet, C. Cauvet, D. Meddahi, F. Semmak, Applying object-oriented analysis on a case study, Inf. Syst. 19 (3) (1994) 199–209.
[3] H. Lee, C. Yoo, A form driven object-oriented reverse engineering methodology, Inf. Syst. 25 (3) (2000) 235–259.
[4] J. Berger, P. Middelraad, A.J. Smith, EURIS, The European railway interlocking specification. UIC, Commission 7A/16, 1992, in: Proceedings of the IRSE 1992/93, 1993, pp. 70–82.
[5] D. Bohus, A.I. Rudnicky, Modeling the cost of misunderstanding errors in the CMU Communicator dialog system, in: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, 2001, pp. 252–255.
[6] M. Bosschaart (M.Sc. thesis), Lean Engineering Design of Rail Interlocking Systems with RailML, Department of Transport & Planning, Civil Engineering & Geosciences, Delft University of Technology, Delft, the Netherlands, 2013.
[7] X. Chen., Y. He, H. Huang, A component-based topology model for railway interlocking systems, Math. Comput. Simul. 81 (9) (2011) 1892–1900.

[8] T. Chiang, H. Hau, H.M. Chiang, S.Y. Kob, C.H. Hsieh, Knowledge-based system for railway scheduling, Data Knowl. Eng. 27 (3) (1998) 289–312.
[9] J. Dorn, Dependable reactive event-oriented planning, Data Knowl. Eng. 16 (1) (1995) 27–49.
[10] N. Fries (M.Sc. thesis), Modellierung einer Eisenbahn-Infrastruktur in RailML, TU Dresden, Dresden, 2003.
[11] G. Geng, L.X. Li, Scheduling railway freight cars, Knowl. Based Syst. 14 (5) (2001) 289–297.
[12] D. Hürlimann, V.P. Krauss, RailML – einheitliche datenschnittstellen für eisenbahnen. in: VWT, vol. 19, Dresden, 2003.
[13] M., Lehmann, T. Albrecht, Erarbeitung eines XML-basierten Schemas zur Darstellung der sicherungstechnischen Streckenausrüstung in einem Fahrsimulator, Institut fur Verkehrstelematik, Dresden University of Technology, 2008.
[14] C. Linder, M. Grimm, Datenmodellanalyse zum Austausch von Projektierungsdaten fur Stellwerkssysteme in INESS, Signal und Draht 104 (9) (2012) 16–21.
[15] A. Nash, D. Huerlimann, J. Schuette, V.P. Krauss, RailML – a standard data interface for railroad applications, Comput. Railw. 9 (2004) 233–242.
[16] RailML organization: ⟨http://www.railml.org⟩, 2014 (accessed 01.05.14).
[17] B. Reuter, J. Rohde, Enterprise application integration, in: H. Stadtler, C. Kilger (Eds.), Supply Chain Management and Advanced Planning, Springer-Verlag, Berlin, 2007, pp. 257–259.
[18] A.F. Rio, M. Rasoamanana, and P. Dumont, Method and apparatus for generating and editing a diagram of railway signalling, European Patent Office, A.T. SA, Editor. Belgium: 2012.
[19] E. Roanes-Lozano, L.M. Laita, An applicable topology-independent model for railway interlocking systems, Math. Comput. Simul. 45 (1–2) (1998) 175–183.
[20] E. Roanes-Lozano, E.R. Macias, L.M. Laita, Railway interlocking systems and Gröbner bases, Math. Comput. Simul. 51 (5) (2000) 473–481.
[21] M.A. Salido, M. Abril, F. Barber, L. Ingolotti, P. Tormos, A. Lova, Domain-dependent distributed models for railway scheduling, Knowl. Based Syst. 20 (2) (2006) 186–194.
[22] E. Sitzmann, W. Eilers, Beltriebliche untersuchung von eisenbahn-knoten, ETR 39 (11) (1990) 691–698.
[23] H. Stadtler, Supply chain management – an overview, in: H. Stadtler, C. Kilger (Eds.), Supply Chain Management and Advanced Planning, Springer-Verlag, Berlin, 2007, pp. 9–36.
[24] G. Theeg, A. Lykov, Movable track elements, in: G. Theeg, S. Vlasenko (Eds.), Railway Signalling and Interlocking, Eurailpress, Hamburg, 2009, pp. 149–178.
[25] G. Theeg, U. Maschek, O. Nasedkin, Interlocking principles, in: G. Theeg, S. Vlasenko (Eds.), Railway Signalling and Interlocking, Eurailpress, Hamburg, 2009, pp. 61–112.
[26] TU Delft data archive to download the proposed RailML interlocking database: http://www.dx.doi.org/10.4121/uuid:1918e49d-8c0d-410b-b5e8-e8149d30fd68, 2014 (accessed 20.11.14).
[27] D. Wang, X. Chen, H. Huang, A graph theory-based approach to route location in railway interlocking, Comput. Ind. Eng. 66 (4) (2013) 791–799.