System-independent and quality tested availability of railway data across country and system borders by the model driven approach

H. R. Gnägi[1] & N. Stahel[2]
[1]*ETH Zurich, Institute of Geodesy and PhotogrammetryIGP*
[2]*University of Zurich, Institute of Geography*

Abstract

The model driven approach addresses tasks around (geo-) data services by first exactly describing the data to be treated on the system- and format-independent conceptual level. Various services can effectively be supported by the precise conceptual model. We will show how the automatic link to a standard transfer format allows system-independent railway data checking and the successful integration of differently structured track data. A conceptual model for the railway data corresponding to the XML-based transfer format railML© allows the comparison with the automatically generated XML-based model driven format. Different possibilities are presented to combine the two approaches for integrating railway data across system and country borders and providing a sustainable interoperability.

Keywords: Model driven approach, data integration, interoperability, railway data, transfer format, semantic transformation, railML©, UML, INTERLIS

1 Introduction

Our application of the model driven approach to railway data started in summer 2003. The question was, if and how very differently structured administrative and commercial data from various systems could commonly be made available without changing the original systems. Especially needed was the link of these data with the track geometry. Hofmann [4] arrived to link CERES and DfA data and to provide a common visualisation. In the LINKOST project of Swiss Federal Railways SBB [2] and the GeoRail project of UIC [1] N. Stahel and H.R.

Gnägi [7] provided a demonstrator extending Hofmann's data hub. The possibility to inspect the data behind the graphical representation was added to the data hub in a client-server-environment. In the GeoRail project we have as well been confronted with the task to integrate German DB track data into the Swiss SBB system and vice versa given different coordinate systems, data structures and exchange formats on each side. Again the restriction had to be respected not to change any of the two given systems. The solution of this problem mainly on a system- and format-independent level will be treated in this paper (chapter 4).

Data transfer using the model driven approach starts always with an analysis and a system- and format-independent exact description of the structure of the data to be transferred. This description is called conceptual data model or conceptual schema. The transfer format (description) can then automatically be derived from the conceptual schema according to rules – once fixed as standard – by a piece of software called compiler. We have therefore been interested to investigate the data structure behind railML©, an exchange format for railway data described by XML-schema. We provided a conceptual schema of this data structure and automatically derived an XML-transfer-format from it. We will report the results of the comparison of railML© with the automatically derived format (chapter 3).

Chapter 2 gives a short introduction to the model driven approach (MDA) and its actual applications to railway problems, chapter 5 provides an overview of ongoing work and chapter 6 contains conclusions.

2 The model driven approach (MDA)

The model driven approach (MDA) together with semantic transformation applied to the exchange of differently structured data with the same content can shortly be described by the following four phases:

(A) Before any data are reformatted or exchanged or treated by any other service, the structure of these data is exactly described on the system- and format-independent conceptual level. This description by a conceptual schema language is called data model or conceptual schema. A widely used graphical conceptual schema language (CSL) is the unified modelling language (UML).

(B) From this conceptual schema, once tested for syntactical correctness by a compiler, can then automatically be calculated by the same compiler the description of the corresponding standard transfer format according to fixed rules (e.g. the description of an XML transfer format like railML© by the XML-schema language).

(C) The data of the systems concerned are reformatted from the proprietary format of their systems to the standard format corresponding to their conceptual schema, done by a piece of software called 1:1-processor. This is a rather easy task, because the two formats correspond to the same data structure: The original format was the basis for the conceptual schema and from this conceptual schema the standard format has automatically been produced.

(D) Given the conceptual schema and the data for the different systems, there exists a set of system-independent tools e.g. to automatically check the quality of the data (a checker can compare the data with the corresponding conceptual schema), for data transfer even between differently structured systems (by conversion systems allowing the mapping of data models and by existing 1:1 processors for different standard data formats) or – as side effect – for a well documented data save (data model plus data in corresponding standard format)

Phases (A) and (B) are part of the core of the MDA corresponding to the classical four shell modelling procedure of the data base design. From the description of the reality selection by natural language, taxonomies and thesauri (shell 1) is developed the conceptual schema (or conceptual model, shell 2, this is our phase A). The conceptual schema has to be transformed to the possibilities of an actual data base according to its logical elements, with the logical schema as result (or logical model, shell 3). In the case of data base or GIS generation, the physical level is normally no more described by a physical schema (or physical model, shell 4) but automatically implemented by the data base software given the logical schema. Quite different is the case of data exchange: There no logical schema is needed but a physical or format schema (or format model, shell 4, our phase B) describing the transfer format. This format schema should automatically be derivable from the conceptual schema.

Phase (C) builds the bridge between the given mostly proprietary format and the needed standard format for the starting data, and vice versa for the target data.

Phase (D) allows – among others – the restructuring of data with the same content by mapping different conceptual schemas. We have used this option to integrate railway data. This so called semantic transformation needs not only the conceptual schemas of the data structures concerned and the mapping of them, but also the data in a standard format corresponding to the conceptual schemas.

We are using the following tools and transfer formats in the four phases:

(A) The conceptual schema languages UML (graphical, for overviews) and INTERLIS 2 (textual, object-oriented, for precising data types, constraints etc.), the UML-INTERLIS editor and the INTERLIS compiler [5]

(B) In the railML© chapter 3 we use the format INTERLIS-XML. In the transformation and transfer chapter 4 we use INTERLIS Transfer Format, ITF. The following format descriptions are automatically available from the INTERLIS compiler:

Format	Format Description Language
INTERLIS Transfer Format ITF	fmt-File (proprietary)
INTERLIS XML	XML-Schema
GML (Geography Markup Language)	XML-Schema

(C) We had to write the 1:1 processors ourselves, because the very special railway formats we had to deal with are not yet treated by standard software.
(D) For semantic transformation we used the INTERLIS conversion system ICS.

We will apply in chapter 3 the phases (A) and (B) to compare the manually developed XML-schema description of railML© with the XML-schema description automatically derived from a corresponding conceptual schema. In chapter 4 the whole scale of phases (A) to (D) will be used to exchange railway data.

3 The railML© transfer format automatically generated from a conceptual model

The initiative railML.org started in 2001 because of difficulties to link different railway software. railML© (railway markup language) is a common evolutionary project of railways, software and consulting enterprises as well as international research institutions. The Fraunhofer institute for traffic and infrastructure systems (IVI) in Dresden (Germany) is administrating and coordinating the partnership [6]. railML© is an XML-based exchange format, whose description (the physical or format schema, see chapter 2) is given in the description language XML-schema.

```
<?xml version="1.0" encoding="UTF-8"?>
.....
  <!--railML: element definitions: -->
  ......
  <!--Infrastructure: track data: -->
  <xsd:complexType name="trackType">
    <xsd:sequence>
      .....
      <xsd:element name="trackTopology" type="trackTopologyType">
      </xsd:element>
      .....
    </xsd:sequence>
    <xsd:attribute name="trackID" type="xsd:string" use="required">
    </xsd:attribute>
    <xsd:attribute name="type" use="optional">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="mainTrack"/>
          <xsd:enumeration value="secondaryTrack"/>
          <xsd:enumeration value="connectingTrack"/>
          <xsd:enumeration value="sidingTrack"/>
          <xsd:enumeration value="stationTrack"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="mainDir" type="dirValidityType" use="optional">
    </xsd:attribute>
    <xsd:attribute name="trackName" type="xsd:string" use="optional">
    </xsd:attribute>
  </xsd:complexType>
......
</xsd:schema>
```

Figure 1: XML-schema of railML©

From the point of view of the MDA this development is located in phase (B) and it would be interesting to have a conceptual description of the data "behind" the transfer format, as provided by phase (A). Therefore we started with the XML-schema description of the format (see figure 1) on the basis of a data example (corresponding part of XML-data in figure 2).
Analysing the XML-schema description of railML© and taking into account the data example, we have been able to get a graphical conceptual schema as UML-diagram (see figure 3).

```
<?xml version="1.0" encoding="UTF-8"?>
<railml xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns=www.railml.org/schema/infrastructure/v100
 xsi:schemaLocation="http://www.railml.org/schema/infrastructure
 /v100 infrastructure_v100.xsd">
  <infrastructure version="1.00">
    <lines>
      .......
      <line lineID="DA" lineName="D-C-B-A">
        <infraAttrGroupID>1</infraAttrGroupID>
        <lineDescr>Line D-C-B-A: Mainline</lineDescr>
        <tracks>
          ......
          <track trackID="D-1" type="stationTrack"
           trackName="Station D; Stationtrack 1" mainDir="both">
            <trackDescr>Station D; Stationtrack 1</trackDescr>
            <trackTopology>
              <trackBegin>
                <bufferStop elemID="StartTrackD-1" pos="0.000"
                 absPos="84.250"/>
              </trackBegin>
              ........
              <crossSections>
                <crossSection pos="0.125" absPos="84.125" ocpIDRef="D"
                 ocpTrackID="1"/>
              </crossSections>
            </trackTopology>
          </track>
          .......
        </tracks>
      </line>
    </lines>
  </infrastructure>
</railml>
```

Figure 2: RailML© data corresponding to the XML-schema of figure 1

A main problem of this first part of phase (A) was to get an object-oriented overview of the heavily tagged and deeply nested XML-schema text. In the second part of phase (A) the INTERLIS-UML-editor provided automatically from the UML-diagram a skeleton of the corresponding textual conceptual schema in INTERLIS (2) CSL. We completed this draft INTERLIS schema by attribute types, especially by geometric ones, by value constraints etc. From this precise conceptual schema the INTERLIS compiler is able to automatically derive the description of a corresponding transfer format. In figure 5 is shown the part of the XML-schema description of the INTERLIS-XML format corresponding to the XML-schema of the railML© format of figure 1.

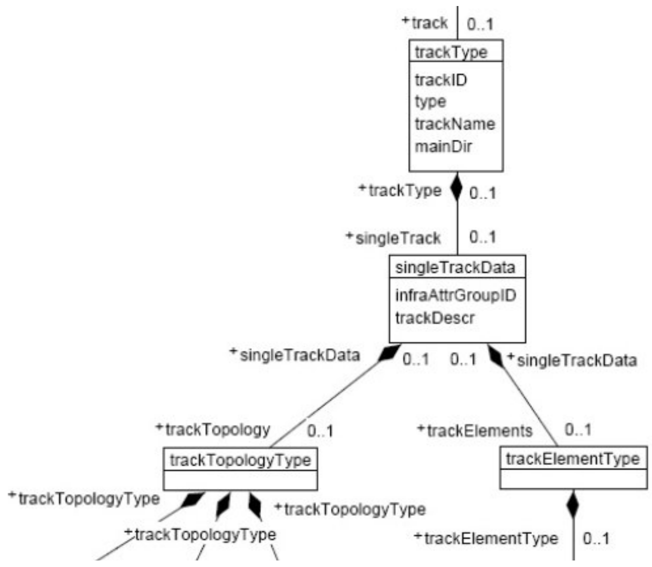

Figure 3: Part of the graphical conceptual schema for railML© as UML-diagram

```
INTERLIS 2.2;
MODEL railML =
!! Model corresponding to railML/infrastructure
  DOMAIN
    dirValidityType = (none, up, down, both, unknown);
    .....
  TOPIC infrastructure =
    .....
    STRUCTURE singleTrackData =
      .....
      trackTopology: trackTopologyType;
      .....
    END singleTrackData;
    STRUCTURE trackType =
      trackID: MANDATORY TEXT*20;     !!Unique track ID
      type: (mainTrack,secondaryTrack,connectingTrack
            ,sidingTrack,stationTrack);
      trackName: TEXT*20;
      mainDir: dirValidityType;       !!Main direction of mileage
      singleTrack: singleTrackData;
    END trackType;
    .....
  END infrastructure;
END railML.
```

Figure 4: Part of the textual conceptual schema for railML© in INTERLIS (2) CSL

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.interlis.ch/INTERLIS2.2"
 targetNamespace="http://www.interlis.ch/INTERLIS2.2"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
.....

    <xsd:complexType  name="railML.infrastructure.singleTrackData">
      <xsd:sequence>
        ....
        <xsd:element name="trackTopology" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="railML.infrastructure.trackTopologyType"
               type="railML.infrastructure.trackTopologyType"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        .....
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType  name="railML.infrastructure.trackType">
      <xsd:sequence>
        <xsd:element name="trackID">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="20"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="type" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="mainTrack"/>
              <xsd:enumeration value="secondaryTrack"/>
              <xsd:enumeration value="connectingTrack"/>
              <xsd:enumeration value="sidingTrack"/>
              <xsd:enumeration value="stationTrack"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="trackName" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:maxLength value="20"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="mainDir" type="railML.dirValidityType"
         minOccurs="0"/>
        <xsd:element name="singleTrack" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="railML.infrastructure.singleTrackData"
               type="railML.infrastructure.singleTrackData"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
```

Figure 5: XML-schema for the INTERLIS-XML transfer format corresponding to the XML-schema for railML© of figure 1

Comparing the XML-schema automatically derived from the conceptual model and the original XML-schema, which was the basis for the definition of the conceptual model, we can remark the following issues:

- The main structure of the two XML-schemas is in principle the same.
- INTERLIS-XML uses mainly XML-elements to encode conceptual attributes whereas railML© works primarily with XML-attributes. XML-elements are easier extendible whereas XML-attributes could be faster treated.
- In INTERLIS-XML all inherited elements are explicitly included in the specialised class whereas in railML© they are only cited by the keyword `extension base`. Because this citation possibility is not supported by all XML-tools, the designers of INTERLIS-XML decided to use the inclusion.
- INTERLIS-XML uses at different places `mininclusive/maxinclusive value` instead of `total/fraction digits`. This provides a more exact basis for range checks.
- From the point of view of range checking maximal text lengths – as necessary for the actual INTERLIS version – are an advantage too.

There exist several possibilities to overcome these differences between INTERLIS-XML and railML©.

- A 1:1 processor railML© ↔ INTERLIS-XML can be developed.
- The INTERLIS compiler can be completed: Beside the format descriptions for INTERLIS-XML, for ITF and for GML the XML-schema for railML© could as well be produced according to a given conceptual schema.
- Finally the XML-schema of railML© can be described as exactly as possible on the conceptual level by UML/INTERLIS. The corresponding XML-schema for INTERLIS-XML is automatically derived and used as new XML-schema for a revised railML©.

This shows that there exist different possibilities to open the door to the advantages of the model driven approach for the railML© initiative.

4 Exchange of railway data across country borders

One result of our demonstrator work in the GeoRail project was the restructuring of German (DB) track data to a (reduced) Swiss track data structure including the coordinate transformation to ETRS. Then we arrived to visualize the track geometry based on DB- and SBB-data at the border between Germany and Switzerland in the region of Schaffhausen ([1], chapter 5.9 to 5.11). For the graphic presentation in the GeoRail demonstrator it was sufficient to arrive at the same (simplified) presentation structure. Whereas for the data exchange it is necessary to exactly describe the start and the target data structure as well as the mapping between them.

The detailed analysis of the German and the Swiss track data showed that in principle the tracks form a graph with nodes and edges. Now this graph and especially its edges can be defined in very different ways:

- Either by a class of nodes of order 3 at least, i.e. with at least 3 edge-lines starting and/or ending at this node, and a class of edge-lines connecting 2 nodes. The track elements of a certain connection type (straight line segment, circle arc, clothoid etc,) are integrated in the edge-lines.
- Or the track-lines are "atomised" into track elements of a certain connection type (see before) and the edges are formed by the belonging track elements. The class of nodes contains then nodes of order 2 at least.
- Or there is one class of nodes and the edges are distributed over different classes of track-lines and/or track-elements as described in the two other cases.
- etc,

We will now clarify the possibilities of data structure description (modelling) and data structure transformation (mapping) by looking at the first and the third of the track data models described above. The first is somehow a pure network model, will therefore be called "PureNet* and could possibly deserve as core of a UIC reference model. The third example provides as another model a relatively complex track element based model, therefore called "TrEl", which is partly inspired by the actual German and Swiss track data.

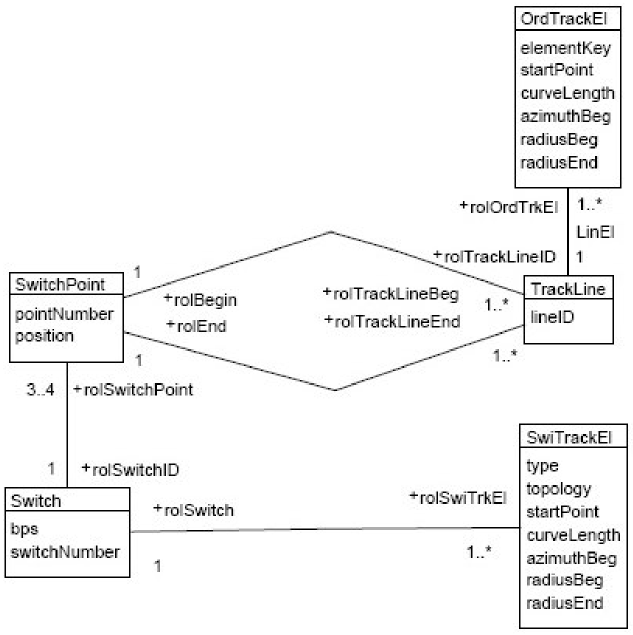

Figure 6: Conceptual schema TrEl as UML-Diagram

Phase (A) of the MDA provides the two graphical conceptual schemas in UML as figure 6 and figure 7. The complexity of the TrEl model has to do with the fact that track lines are subdivided into ordinary track elements and into switch track elements. The ordinary track elements are linked by the start- and end-point associations to switch points. But the switch track elements are directly associated to the switches. A switch object itself is a composition of 3 or 4 switch point objects (as can be seen in figure 6). Therefore it will not be easy to collect all track elements of a track line in the TrEl model.

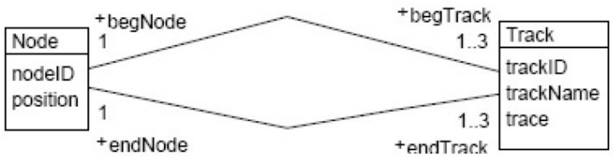

Figure 7: Conceptual schema PureNet as UML-Diagram

The PureNet model on the other side is much simpler, because it consists only of the classes node and track. The nodes correspond to the (TrEl) switch points. The tracks are the glued collections of (TrEl) track elements (ordinary and switch ones) between two nodes described by the attribute `trace`, which is of type `POLYLINE`. Additional attributes have to be added in the PureNet classes to not loosing the switch track element properties of the TrEl model.

The corresponding textual conceptual schemas are using the textual conceptual schema language INTERLIS 2. The precise specifications of the data types are now given for every attribute. In the two models we have two different types for modeling the same geometric property. For track elements (of ordinary and of switch type) the geometry is defined by the coordinates of the start point, the element length along the curve, the azimuth of the curve tangent in the starting point, the radius in the starting point and the radius in the end point. With PureNet the whole trace of a track is defined by the data type `POLYLINE` which has straight line segments, circle arcs, and clothoid segments (more types exist) as possible connection geometries of a single part (track element) of the whole track.

```
MAPPING MODEL TrEl2PureNet =
  ...
  TRANSFORMATION GROUP Group1 =
    SOURCE { TrackLine, OrdTrackEl, ASSOCIATION LinEl}
    TARGET { Track }
    MAPPING {
      RULE TrkLine2Trk { Track := TrackLine; }
      RULE Glue { setPOLYLINE(LinEl, Track.trace); } }
  END Group1;
  ...
END TrEl2PureNet;
```

Figure 8: Mapping two models

In phase (D) the mapping from the `TrEl` model to the `PureNet` model has to be defined. Figure 8 shows one of different transformation groups as part of this mapping model. The two classes `TrackLine` and `OrdTrackEl` of the Model `TrEl` (see upper right part of figure 6) are transformed into one class `Track` of the model `PureNet` (on the right of figure 7). In the transformation group `Group1`, `SOURCE` defines the classes and associations of the source model `TrEl` used by this transformation group, and `TARGET` indicates, that class `Track` of the target model `PureNet` will be produced. The keyword `MAPPING` initiates the list of rules defining the transformation. The `RULE Trackline2Trk` defines, that for each instance of the class `TrackLine` is produced an instance of class `Track` taking over attributes with same name and same data type (here no one). The second `RULE Glue` is the start of the general method `setPOLYLINE` taking into account all the `OrdTrackEl` objects linked to the actual `TrackLine` object by the association `LinEl` to produce the POLYLINE-geometry of the attribute `Track.trace`. The INTERLIS-like definition language for map description used in figure 8 is actually under development (see also H.R. Gnägi, A. Morf, P.Staub [3]). The declarative definition of the maps will allow different techniques for the implementation. At the moment this mapping language needs to be translated into one of the implemented mapping systems mentioned in chapter 2 (like ICS or FME).

As described in chapter 2 the semantic transformation defined above needs not only the conceptual schemata of the models in question and the mapping definition but in addition also the data to be transformed in a standard format corresponding to the model. Because the selection of Swiss data we can use as example of the start model `TrEl` is in a special proprietary format, a special 1:1 processor had to be programmed in phase (C). Its result is used as input to the semantic transformation, whose output is again in the standard format. With a checker tool the result of the 1:1 processor as well as the result of the semantic transformation can be tested against the conceptual schema. With this quality checks the restructuring process ends. It starts with the system `TrEl` and results in data according to the system `PureNet`. The whole chain of system-neutral tools used without modification of the start and target GIS will be shown in a live presentation at CompRail 2006.

5 Further work

The differences in the modelling principles of railML© (data type nesting) and of the MDA (class and association graph) need a deeper analysis. The linking possibilities between the two methods mentioned at the end of chapter 3 should be seriously compared to be able to evaluate, which strategy provides the optimal synergy effect with the minimal effort. By looking at the details of railML© we asked us the following question: Can the railML© data structure be simplified without loss of information? We would like to answer this question. To gain more experiences about methods and structures other railway application areas beside infrastructure will be treated with the MDA. Clearly there is an urgent need to implement the conceptual mapping language used in chapter 4.

6 Conclusions

We have presented two approaches to railway data and their exchange. On one side is the detailed collection of huge railway knowledge at the XML-format level by the railML© initiative. On the other side the system- and format-independent model driven approach offers among others the obvious advantages of automatic quality testing and of easy data restructuring without need for system restructuring. A successful combination of the two methods is conceivable and seems to provide a solid basis for a sustainable interoperability and integrability of railway data across Europe.

7 References

[1] Engel, T., Barbu, G., Gnägi, H. R., Lahr, B., Müller, S., Robert, D., Stahel, N., Winter, P., *Gleisbau und Gleisunterhalt auf der Basis absoluter Koordinaten. Schlussbericht des UICProjektes Georail.* SBB, Bern, 15.5.2005

[2] Engel T.: Barbu, G., Gnägi, H. R., Winter, P., Georail, presentation, NAVSAT, Geneva Switzerland, 23rd June 2003.

[3] Gnägi H.R., Morf A., Staub P., Semantic Interoperability through the Definition of Conceptual Model Transformations. *Proc. of the 9th AGILE International Conference on Geographic Information Systems,* to be published 2006.

[4] Hofmann R.: *Modellbasierte Datendrehscheibe für die SBB*, Bericht Vertiefungsblock, ETH Zürich D-Baug IGP-GF, 2003

[5] INTERLIS 2, reference manual, user manual, tools. www.interlis.ch

[6] railML© initiative, www.railml.org

[7] Stahel N., Gnägi H.R., Georail Demonstrator. Presentation, UIC-Georail-Phase II: CNTD and ETRS, Workshop Paris, 15th September 2004