RailML – a standard data interface for railroad applications

A. Nash[1], D. Huerlimann[1], J. Schuette[2] & V. P. Krauss[2]
[1]*Institute of Transportation Planning and Systems, Swiss Federal Institute of Technology, Switzerland*
[2]*Fraunhofer Institute for Transportation Systems and Infrastructure, Germany*

Abstract

The growing number of computer programs used to model different aspects of railway operations has created the need for a simple and efficient way to transfer data between applications. In the past, specialized interface programs have been developed to transfer data but this is an inefficient and time-consuming process. As the number of different railway simulation and operations programs increases, developing and maintaining individual interfaces will become impractical. RailML has been developed using the XML (eXtensible Markup Language) to simplify data transfer through the use of a common data structure. Programs using the RailML language produce export files with the RailML structure, these files can then be used directly by other RailML compatible programs. RailML is an open source project being supported and developed by a growing list of partners. The paper presents a more detailed description of RailML, outlines the application's current status, and invites new partners to join the consortium.
Keywords: railway simulation, timetabling, XML, RailML, data transfer, transportation data mark-up languages.

1 Introduction: problem and solution

RailML is an open source data structure that has been developed to simplify the transfer of data between various railroad simulation and operations computer programs. This chapter describes the RailML program and the problem it has been developed to address.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

1.1 Railway data exchange problem

Today, the computer has become a standard tool in the planning and operation of all businesses, and railroads are no exception. In the last few years the number of specialized computer programs designed to address all aspects of the railroad business from timetable construction to staff assignment to infrastructure planning has mushroomed. In most cases these programs have different data structures and protocols. As the number of programs and applications grows, the variety of different data structures becomes more and more unwieldy.

In order to efficiently use several different applications for planning improvements to railroad operations together, data must be exported from one program and imported into another for further processing. This can be done by copying data between applications (a long and tedious process for programs with any significant amount of data) or by using an automatic data transfer program (transparent for the user). Several specialized interfaces between programs currently exist, but as the number of railway applications grows this solution becomes less efficient.

Transferring data between two particular programs requires that two different interfaces be developed (one for each direction of transfer). As the number of programs increases the number of possible data transfers increases significantly. For example, assuming there are five programs, in theory twenty specialized interfaces need to be developed for full data exchange between all programs. In general terms, if the number of programs, *n,* increases linearly, the number of interfaces *[n * (n-1)]* increases quadradically.

As the number of new technical solutions, such as Tracking & Tracing, real time estimates and adjustments, increases, so will the number of railway programs increase. Clearly, the number of new interfaces to be developed for these new applications represents a great challenge to the railroad industry.

1.2 Standard formats for efficient data exchange

The situation in the railroad industry is not unusual, as computers become more ubiqueous in society there is a growing need for efficient data exchange. Today these types of data incompatibility problems are being addressed using "open and universally usable data formats". Such formats are generally developed using the eXtensible mark-up language (XML) as a basis (e.g. MathML for mathematical expressions and GML for geographical data).

XML was developed by the World Wide Web consortium for use in WWW applications (the most well-known mark-up language is XHTML). XML is not an application language, but rather a set of rules that can be used to define other “mark-up” languages and therefore functions as a meta language. The major advantage of XML based documents is that they describe both data as well as the data’s structure. Therefore, XML was an ideal solution for transfer and storage of railroad data.

The data transfer problem created as the number of applications increases can be solved by means of EAI (Enterprise Application Integration). EAI enables the collaboration of different programs by creating standard interfaces, which are

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

independent of any given application, but focus on the objects to be exchanged. The corresponding object descriptions are based on XML syntax.

Effective data exchange between two applications requires that each program include a function for generating XML data (export) and a function for reading in and interpreting XML files (import). Due to both the widespread use of XML and the standardization caused by the World Wide Web, most computer application development platforms include pre-defined libraries and functions for the processing of XML based data, which substantially reduces the time needed to develop import and export functions for creating specialized applications such as for railroads.

1.3 Development of RailML

In 2001, a group of railroad application developers formed a free partnership with the objective of developing an open source XML-based language for railroad applications that would be called RailML. The partnership was originally organized by researchers from the German Fraunhofer Institute for Transportation Systems and Infrastructure in Dresden (FhG-IVI) and the Swiss Federal Institute of Technology's Institute for Transportation Planning and Systems (ETH IVT).

The partnership is organized under the name RailML Consortium (www.railml.org) and has expanded to include researchers from several universities, railroad operating companies, private research institutes, and consulting firms. The Fraunhofer Institute serves as the partnership's technical coordinator providing resources such as the web page and discussion forum.

As a partnership, the development of RailML is based on suggestions, comments, and criticism of the partners. Individual application developers work together in a coordinated fashion to complete the actual work of developing elements of RailML. These developers form groups (open to all interested partners) based on individual interests and then communicate through moderated on-line user forums, newsgroups, and at regular meetings. As groups finish work on RailML elements, consortium members serving as technical coordinators finalize them and publish the standards (similar to the RFC for the Internet).

In summary, RailML standards are developed in the context of technical discussions that are open to everyone interested in developing applications for the railroad industry.

2 RailML schemas

RailML is a generic language that can be used to describe railway-related data. The language has been divided into sub-formats (or schemas) for particular types of railway data. RailML consortium partners are currently developing data structures for the three main types of data files used in railroad simulation: timetable (schedule), infrastructure (with subsets for lines and stations), and rolling stock. Figure 1 illustrates the RailML schemas.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

At present the RailML timetable structure is most developed and has already been incorporated into several applications (e.g. OpenTrack, FBS, Viriato and OpenTimeTable [1]). A significant amount of work has been completed on both the infrastructure and rolling stock schemas, but these have not yet been finalized. Several additional schema have been suggested.

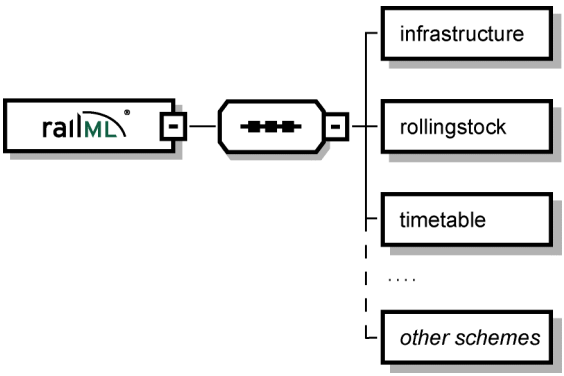

Figure 1: Existing RailML schemas.

As a web-based development partnership, each schema has its own newsgroup open for discussion. The newsgroups' purpose is two-fold: first, newsgroup partners exchange information; second, knowledge and processing of definitions is discussed and archived. RailML.org also supports a newsgroup for general exchange and discussions that are not schema-related.

3 Data transfer with RailML

RailML is a simple and efficient way to transfer data between computer programs used to model different aspects of railroad operations. Programs using the RailML language produce export files with the RailML structure; these files can then be used directly by other programs. The receiving program parses the incoming file to obtain only the data it needs, which allows many different programs to use the same data file.

RailML provides two ways for data exchange. The first consists of running applications separately and having them produce output files that are then used as inputs for other applications. The second approach, currently under development, consists of transferring data between two applications directly by means of interprocess communication (e.g. Web services via TCP/IP) to expenditure-exchange, i.e. without the detour of a file. Figure 2 illustrates the two forms of data transfer process.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

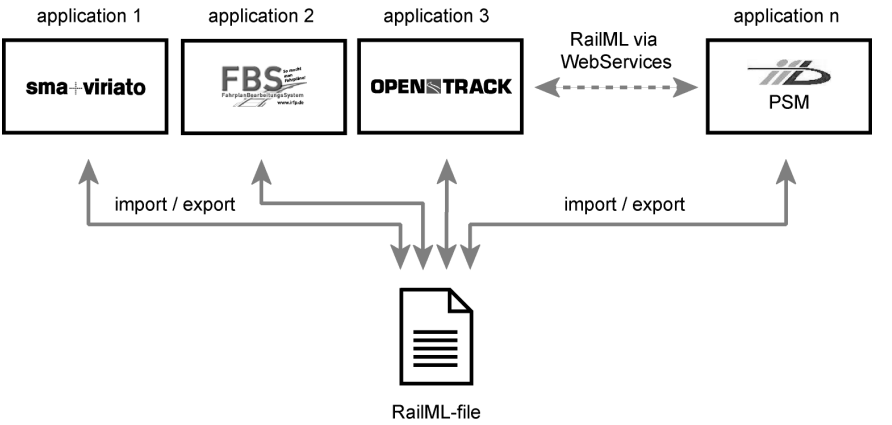

Figure 2: Examples of RailML data transfer.

4 RailML structure

This chapter describes the general structure of a RailML file using the timetable schema as an example.

RailML is an XML-based language, this means that data files include both data and descriptions of the data that they contain. All XML derived languages use a very simple and flexible ASCII title format for their documents. In all cases the documents are hierarchical, they form of a tree layout, i.e. each document has a clear root element, from which navigation can start using the general document structure (common to all RailML documents) as a guide.

The RailML document's root element is called <railml>. The sub-schemas (infrastructure, rolling stock, and timetable), which contain the railway-relevant data, are then derived from this root structure. Using this flexible approach, each individual RailML compatible application determines which particular types of data should be used.

The data in XML documents are organized and administered by means of elements and attributes. A RailML element begins with a start tag "< railml >", and ends with an ending tag "</railml>". An element can have attributes for more detailed description and may also contain additional elements.

Figure 4 illustrates an example RailML document, which contains timetable data. Under the element "< timetable >" are listed the individual courses in the element "< train >". The most important attribute of a course, the course number (trainID), is required; other attributes are optional.

Every train running (course) contains as many timetable entries as desired "< entry >", which are arranged listed under the element of "<timetableentries>". The individual timetable entries contain attributes such as station abbreviation (pos ID), arrival time (arrival), departure time (departure), or minimum station stopping time (min stop time).

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with OpenTrack (http://www.opentrack.ch) -->
<railml xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
  instance" xsi:noNamespaceSchemaLocation="timetable.xsd">
  <timetable version="0.95" scheduleformat="hh:mm:ss"
    periodformat="s">
   <train trainID="RX 100.2" type="planned"
     source="opentrack">
     <timetableentries>
       <entry posID="ZU" departure="06:08:00" type="begin">
       </entry>
       <entry posID="ZWI" departure="06:10:30" type="pass">
       </entry>
       <entry posID="ZOER" arrival="06:16:00"
          departure="06:17:00" minStopTime="9" type="stop">
       </entry>
       <entry posID="WS" departure="06:21:00" type="pass">
       </entry>
       <entry posID="DUE" departure="06:23:00" type="pass">
       </entry>
       <entry posID="SCW" departure="06:27:00" type="pass">
       </entry>
       <entry posID="NAE" departure="06:29:00" type="pass">
       </entry>
       <entry posID="UST" arrival="06:34:30" type="stop">
       </entry>
     </timetableentries>
   </train>
  </timetable>
</railml>
```

Figure 3: Example RailML document.

As Figure 3 illustrates, the RailML language is very similar to other XML-based languages and therefore relatively easy to understand and use.

5 RailML consortium

As outlined above, RailML is being developed as open source data format by a group of railroad application developers who have joined together to form the RailML Consortium. Program development is being led by the Fraunhofer Institute and now includes many partners from academia and industry. As a partnership all development is based on suggestions, comments, and criticism of the partners.

Partners have two main responsibilities; first, to help develop RailML; and, second to encourage the wide dissemination and application of RailML. New partners are welcome to join the consortium at any time [2].

There are two types of partners: Supportive Partners and Development Partners. Supportive Partners receive the copyright for XML schemas (according to the licensing terms and conditions) and are listed on the RailML web site with a link to their respective homepage. Development Partners take an active part in

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

developing schemas by providing suggestions, comments, and recommendations. Each schema has its own set of development partners.

Today the use of the RailML-schemes by all partners is free, as long the formats and the copyright of railML.org are respected. Since autumn 2003, RailML is a registered trademark.

Every application, which is be able to read or write RailML-data by a railML-scheme can be allowed to use the following RailML-Logo:

Figure 4: RailML-logo.

6 Conclusions – benefits for simulation and timetabling

The RailML project's objective is to simplify data transfer between the growing number of computerized rail simulation programs through the use of a common data structure. This is extremely beneficial in that it allows railroad planners and operators to easily use several different types of computer applications, using each to do what it does best. In the past, specialized interface programs were developed to transfer data between applications, but this is an inefficient and time-consuming process. Furthermore, as the number of applications increases, developing and maintaining these individual interfaces is becoming impractical.

Already several different railroad applications include interfaces that use the RailML format to import and export timetable data between applications. These programs include: FBS, Viriato, OpenTrack, and OpenTimeTable.

References

[1] For more information on these programmes please see: www.opentrack.ch www.irfp.de/fbs, www.viriato.ch, www.opentimetable.ch.

[2] All information about RailML can be found at www.railml.org.

[3] Fries, N., 2003, Modellierung und Abbildung einer Eisenbahn-Infrastruktur auf ein Schema der XML-basierten Schnittstelle RailML (Diplomarbeit, TU Dresden, Professur für Verkehrssicherungstechnik, Dresden).

[4] Hengartner, M., 2003, Grafikeditor für RailML-basierte Infrastrukturdaten (Diplomarbeit, ETH Zürich, Institut für Verkehrsplanung und Transportsysteme, Zürich).

[5] Hürlimann, D., 2002, Objektorientierte Modellierung von Infrastrukturelementen und Betriebsvorgängen im Eisenbahnwesen, 2002, Institut für Verkehrsplanung und Transportsysteme, ETH Zürich, Schriftenreihe des IVT, Nr. 125.

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)

[6] Montigel, M., 2002, Time-triggered exchange of train route data between train control systems, in Computers in Railways, edited by J. Allan, R. J. Hill, C. A. Brebbia, G. Sciutto & S. Sone (WIT Press, Southampton), pp. 33–41.
[7] Hoffmann, R., Krauss, V.P., 2001, RailML als Datenaustauschformat für Eisenbahnsoftware (Projektbeschreibungen und -berichte sowie Konzepte, Fraunhofer Gesellschaft e.V., Institut für Verkehrs- und Infrastruktursysteme, Dresden).

Computers in Railways IX, J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto & S. Sone (Editors)