Challenges of Constructing a Railway Knowledge Graph

Stefan Bischof(✉)[0000−0001−9521−8907] and
Gottfried Schenner[0000−0003−0096−6780]

Siemens AG Österreich, Corporate Technology, Vienna, Austria
bischof.stefan@siemens.com, gottfried.schenner@siemens.com

1 Introduction

International railway networks are an important means for passenger and freight transport. Railway software systems often must be supported and maintained for decades. Different systems cover different aspects such as train protection, signalling, infrastructure hardware and software. Usage of different standards and regulations, both international (e.g., European Train Control System) and national, for these aspects leads to a large number of incompatible systems.

A wide variety of use cases would profit from a network-wide unified/integrated access to the available different (legacy) tools and databases. Starting with asset management, topological queries, or checking consistency an integrated data access system, i.e., a knowledge graph (KG), could also be used as a basis for ensuring network-wide compliance with (safety) regulations, operational use cases like planning for maintenance or train scheduling. Furthermore the KG could be a precious resource for data analytics use cases.

Applying standard information integration approaches is not feasible because of the complexity of the data sources. This complexity is aggravated by the fact that the (engineering) tools are permanently under development and several versions of one tool are usually in production in parallel. So the integration itself must also be an ongoing process. To avoid big upfront cost we thus aim for an agile process to iteratively integrate the data as needed by new use cases.

Considering the necessary flexibility and the network nature of railway systems we chose a graph based formalism. To ensure long-term support of our solution, we rely on standardised technologies as much as possible. Our approach is to integrate data from different tools in a KG (see Fig. 1) under a common domain model (ontology), ideally based on an existing industry standard. The tool schemata as well as the instance data are transformed to graphs.

Individual tool data models are mapped to the common ontology as far as necessary and feasible. Lightweight ontology mapping is performed by custom SPARQL Update queries. Instance data is represented in terms of the common domain model by physical transformation or virtual integration. Instance linking is implemented by SPARQL Update queries.

With the resulting KG it is possible to access the integrated data using the common data model as well as the data remaining only in the context of specific

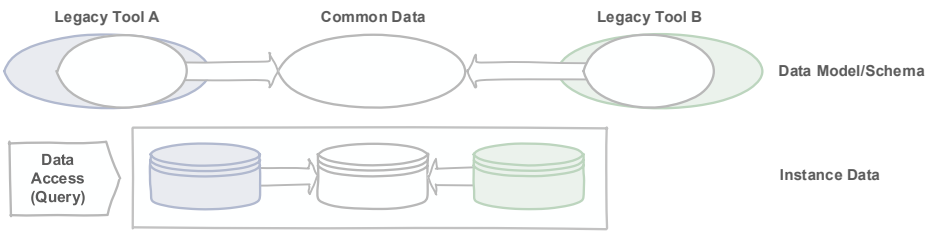

Figure 1. Integration Architecture

(legacy) tools. This architecture enables an agile and iterative integration process where we can integrate data as the need arises.

2 Challenges

When applying our approach on the railway domain we encountered several challenges. Most of the challenges stem from the characteristics of technical systems. Engineering tools often use data models that are highly specialised. Therefore an ontology derived from these data models reflect the system view of the tool, e.g., a tool for hardware configuration will have a total different view of the infrastructure than a tool used for train scheduling. In an industrial KG the correctness and consistency of instance data is of uttermost importance, as faults in the data could lead to accidents. We identified the following main challenges during the construction of our railway KG.

2.1 Lack of Existing Standard Railway Ontologies

To maximise interoperability and possible long-term support of our common domain model, a railway ontology based on an international standard would be preferable. The existing ontologies, mainly being developed in research projects, are usually not openly available and more importantly not built for reuse.

The best candidate for an internationally standardised railway domain model seems to be the RailML XML Schema specifications. We map the relevant type definitions to an ontology which we then use as common domain model in our architecture. Because of the impedance mismatch between XML and RDF as well as the unsatisfying state of automatic mapping tools, the mapping process is a cumbersome and error-prone task.

2.2 Ontology Alignment

Ontologies derived from technical tools are highly use case specific. In contrast to ontologies for biological systems the number of concepts is usually relatively small but the number of data and object properties is much higher. Automatic ontology alignment is infeasible due to different domains (e.g., hardware, software, communication), different levels of granularity, differences in the modelling approaches, and language (German/English).

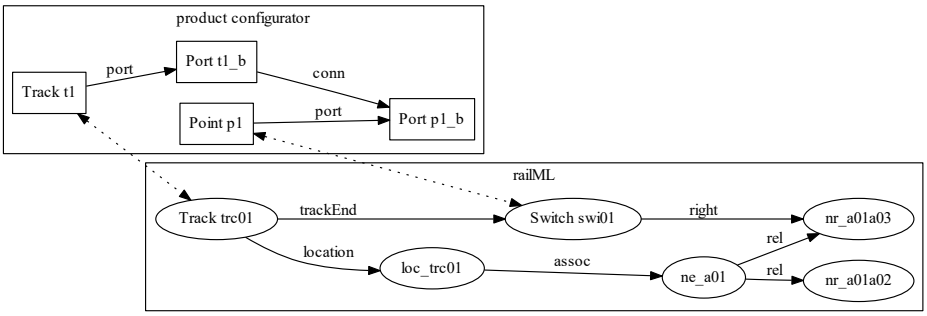

Figure 2. Example Matching of Railway Switch.

2.3 Complex instance matching

Instance matching requires a detailed knowledge of the underlying ontologies. Figure 2 shows the matching of instance data from two different ontologies. The example consists of a railway track that is connected to the right branch of a railway switch. Although there exists a 1-1 mapping (dotted-arrow) between the high level concepts, the determination of the topology of the instances is complex. In the ontology of the legacy tool (in this case a product configurator) the right branch is by convention port "b" of the switch element, whereas in the railML ontology the right branch is defined by linking the switch to the micro topology view of railML. These complex matchings are out of the scope of standard matching tools [3].

2.4 Open and Closed World Reasoning

Technical systems often require to switch between closed and open world reasoning. For example a configurator for interlocking systems will apply closed world reasoning to configure one specific interlocking system. Sometimes partial information of neighbouring interlocking systems is required. These are not completely represented and therefore open world reasoning is applied. The challenge for a KG based system is to apply the right techniques depending on the required reasoning.

2.5 Brownfield Software Development

Brownfield (software) development describes the further development of software in the presence of an existing legacy tool landscape. The challenge is to gradually improve the system while keeping the overall system working. If a KG is introduced into the tool landscape we can not expect that all tools are adapting their knowledge representation accordingly.

The overall management of a railway system involves a lot of different tools. The characteristics of these tools (knowledge representation, programming language, database, data exchange) often reflect the state-of-the-art at the time of

their creation. The programming languages range from shell scripts, imperative, object-oriented programming languages to the web-based frameworks currently en vogue. Data exchange between tools is often realised on a tool to tool basis with a proprietary interface definition. Only recently standards like RailML have been defined to facilitate a standardised way of data exchange [1].

When introducing a KG as a first step it is sufficient to extract as much information as possible from the existing data sources and store it similar to a data warehouse approach. This way the consistency of the data can be checked in a unified manner, data analytic tasks are easier and formal reasoning about the properties of the overall system is made possible [2].

The most challenging task, but in our view also the main benefit of a KG, would be to gradually replace legacy tools with tools that operate on the KG, and in this process reduce redundancies in the data. This would also require the tools to reverse the direction of information flow. Instead of extracting information from the tools for the KG the tools should use the KG to persist their data.

3 Conclusions

The use of graph-based formalisms promises great potential when data from several heterogeneous systems of the same domain should be integrated. Currently we are using the system with data provided from product configurators to answer customer queries which were previously either infeasible or at least involved a lot of manual work. Nevertheless we identified several challenges when implementing a flexible integration architecture with graphs. Unfortunately major efforts will be necessary to fully address these challenges.

For the future we hope that graph based data integration architectures can enable us to practically exploit the potential distributed and hidden in our tools and databases. Particularly lifecycle management (ALM, PLM) could benefit from such a KG derived from engineering data. For an upcoming asset management use case we leverage the KG as a data source. Furthermore, by minimising data redundancy we expect to reduce inconsistencies between different tools and aim to streamline engineering processes.

References

1. Bosschaart, M., Quaglietta, E., Janssen, B., Goverde, R.M.: Efficient formalization of railway interlocking data in RailML. Information Systems **49**, 126–141 (2015). https://doi.org/10.1016/j.is.2014.11.007
2. Luteberget, B., Johansen, C., Steffen, M.: Rule-based consistency checking of railway infrastructure designs. In: International Conference on Integrated Formal Methods. pp. 491–507. Springer (2016). https://doi.org/10.1007/978-3-319-33693-0_31
3. Schenner, G., Bischof, S., Polleres, A., Steyskal, S.: Integrating distributed configurations with RDFS and SPARQL. In: Configuration Workshop. pp. 9–15 (2014), `http://ceur-ws.org/Vol-1220/02_confws2014_submission_3.pdf`