Ontology-driven Data Integration for Railway Asset Monitoring Applications

Jonathan Tutcher
Centre for Railway Research & Education
University of Birmingham
Edgbaston, UK
Email: j.tutcher@bham.ac.uk

Abstract—As the extent to which information systems are used across rail and transportation networks continues to grow, huge potential for data driven decision support and analysis is emerging. Interoperability between systems in these industries is currently poor, and opportunities for such analysis are often missed through unavailability of data. Semantic data modeling provides a mechanism for facilitating greater interoperability between systems, and allows easier integration of data from heterogeneous sources.

This paper describes the current state of the art in rail data modeling, and introduces a asset monitoring system based on contemporary ontology and linked data (semantic data modeling) technologies. The design and implementation of Asset Monitoring As A Service (AMaaS) is shown, and overviews of key design patterns used to ensure extensibility and interoperability given. Finally, the potential for re-use of the system is discussed, along with known limitations and known technology advances. An outline of further work is provided, including in designing methodologies to foster uptake semantic data models across the industry.

I. INTRODUCTION

The UK’s railway network is the oldest in the world, and has been steadily becoming more popular over the last twenty years, with passenger figures rising by 67% from 1991 to 2011 [1]. As renewed investment in infrastructure and rolling stock strives to increase capacity, reliability, and safety, the number of information systems being commissioned across the country continues to grow. Ever-increasing volumes of data acquired from these systems not only provide new opportunities for better decision-making and analysis, but also present huge challenges to traditional approaches of data storage and management. The ever-changing and heterogeneous nature of data sets present across different subsystems make conventional integration techniques such as data warehousing difficult to implement, and these techniques often require a large amount of commitment and resource from operators to realise—for instance in Network Rail’s DARWIN system. New methods of big data analysis are capable of providing powerful business insights, but still require complete and structured data to operate effectively. It is agreed across the industry that new approaches to information management are required to overcome the ever-increasing complexity in rail data [2], and central to this approach is the notion of a common, industry-wide data model.

One application for such data models is in the field of remote rail asset management and condition monitoring. Condition monitoring and predictive/model-based maintenance of railway assets can significantly lower maintenance cost by reducing unnecessary inspections and diagnosing incipient faults prior to equipment failure [3], and many systems are being introduced in the UK to continuously monitor both infrastructure and rolling stock for such faults [4]–[6]. Detection and diagnosis systems have benefited from significant research and improvement in recent years, but usually rely upon fixed data storage and management solutions to operate. Network Rail’s Intelligent Infrastructure project in the UK, for example, makes use of the MIMOSA-CBM [7] standard for data communication and a proprietary Supervisory Control And Data Acquisition (SCADA) system for storage and management. This solution is effective, but the system must be modified significantly for every new asset to be monitored, and new interfaces must be created whenever information sharing between systems is deemed necessary.

When presented with these challenges, the motives for defining a common cross-industry data model are clear—if all information systems can be defined to use common vocabularies and communicate in the same way, interfaces between such systems become simple, and information sharing much easier. These motives have led to the creation of several domain-specific models [8] [9], but long system lifecycles present across the industry have until now inhibited buy-in to one particular tool or standard.

This paper describes implementation of a railway asset monitoring system based upon semantic data models, which offer greater capabilities for data integration, extensibility, and compatibility over traditional approaches. Section II outlines the current state-of-the-art of asset management and information sharing in the rail industry, before section III introduces the demonstration system designed and implemented. Section IV describes more thoroughly the mechanisms used in order to achieve the usability required from our semantic data model, and shows the live system itself. Finally, section V outlines alternative approaches and next steps for the project, as well as lessons learnt.
II. Asset Management and Information Sharing in the Rail Industry

A. Rail System Heterogeneity in the UK and Europe

The UK rail network has endured a number of different methods of governance as both a publicly and privately-run entity since the first modern railways in the world were built in the early 19th century. As such, the industry faces unique challenges, and these challenges provide business cases for new data integration projects not present elsewhere.

Initially, private investors built bespoke railway lines using different standards, technologies, and track gauges, and each route was managed independently. There was little incentive for interoperability at the time, and different companies often operated competing routes between the same destinations. In 1948, when the railways were bought and nationalised under British Rail, routes existed between most major towns and cities in the UK, and a major shift from road to rail was anticipated.Whilst government investment allowed some standardisation of the infrastructure and network, rail transport did not prove popular, and led to the axing of thousands of miles in the early 1960s. By 1994, most of the network operated to newly created British Rail operations standards, but drives to improve efficiency and reduce cost led government to privatise the industry once more.

Since then, the UK rail industry has been operated by companies fulfilling a number of roles. Rail infrastructure is operated and maintained by state-owned Network Rail, passenger and freight services run by several private Train Operating Companies (TOCs), rolling stock owned and leased to TOCs by other organisations (mostly private banks), and safety and governance overseen by the state-funded Railway Safety and Standards Board. This separation of responsibilities between many different stakeholders has led to silofication of information systems, as individual companies develop solutions for their own needs. Interfaces between mainline railways and metro systems (primarily in London) are also complex.

Whilst the situation in Europe is slightly different, the same problems occur. Although some countries operated vertically-integrated systems allowing one organisation to oversee all aspects of rail operation, difficulties arise where services travel across country boundaries. Rail operators are free in the EU to run services across borders without stopping, but differing system capabilities mean that often locomotives must be swapped or passengers must change trains completely. Trains running services across multiple countries may also be fitted with multiple train control systems to interface with each country’s signalling infrastructure, increasing complexity and reducing reliability.

B. Existing Rail Domain Data Models

A number of research projects and industrial initiatives concerning knowledge management and data modeling for railway data have been undertaken over the last decade, aiming to allow better integration of data between systems. Few have enjoyed significant commercial uptake, although support for RailML, a cross-industry project establishing comprehensive XML data models for information exchange, continues to grow in Europe [8]. Other relevant models include efforts by the UIC into a new infrastructure model [10], and the EU FP7 InteGRail Project [11] which delivered a basic rail ontology—an semantically richer graph-based representation of domain concepts and relationships. Many other transportation data models also exist and are widely used; most notably NaPTAN [12] and the Esri ArcGIS model [13].

EU interoperability legislation, coming into force across the railway industry from 2015, will also provide incentives for companies to consider novel methods of data management. The EU Register of Infrastructure (RINF) requires that all railway infrastructure operators across Europe provide a basic level of information about their networks [14], a task which has so far caused many companies difficulty. As European interoperability is mandated further, the demand for efficient data management and exchange is likely to grow.

C. Disincentives for Information Sharing

Common to a number of other industries, organisations within the rail industry often face motives for keeping information they generate to themselves. To train operating companies and suppliers, information is a huge business asset, and seen as a significant commercial sacrifice to share openly. In the UK, many rail maintenance and reporting systems are operated as services, and reports sold back to interested stakeholders—the sharing of raw data for interoperability here may involve giving a substantial amount of their offering away for free. In other cases, common with difficulties faced in the domain of open data on the world wide web, many businesses perceive little or no added value to allowing others access to their system data. In the rail industry, it is likely that a move towards data sharing will be mandated across the industry, as discussed earlier.

D. Semantic vs. Syntactic Data Models

Traditional approaches to data exchange rely upon data formats which store information context in document syntax. Relational databases, XML models, and document stores are examples of this. Whilst these work well for systems which are well-defined and do not change, extending them to facilitate new features can be difficult, and sharing data in multiple formats is not natively possible. Only limited machine understanding of data is possible, since its meaning relies entirely on a structure which is pre-defined by schema documents. Newer ‘NoSQL’ approaches go some way to tackling the first of these problems [15], but still rely upon bespoke system interfaces to be created when information exchange is required.

An alternative to this reliance on document syntax to communicate meaning is through semantic interoperability. Rather than using a pre-defined template or schema to store data, semantic models build a series of assertions about concepts (real world objects or abstract things), which together form a graph of knowledge. Data semantics are built up through creation of links between entities, which unambiguously describe each
piece of data. In Resource Description Framework (RDF), the W3C standard for semantic web knowledge representation, each concept is referenced by a unique identifier, and these identifiers are linked together using triples (facts). Triples link a subject entity with an object entity via a predicate, or relation. As these relations are also user-defined entities, data representation can be very flexible. Examples of triples are shown in Table I.

Additionally, RDF itself is technology-agnostic, and thus data is preserved into the future as new technologies and tools come about. The specification only calls for the representation of facts as explained above, and these facts can be serialised in several different ways. Whilst there is still a dependency on the vocabulary and design patterns used when representing data, the meaning of the data is preserved through time, and so its use as a knowledge representation technique fits well with long system lifecycles present in the rail industry.

### E. Ontologies

In computer science, an ontology is a way of formally describing domain knowledge within an information system. Ontologies can allow the enrichment of data within a given data store by exploiting facts about how the world works to infer new information about each piece of data. The Web Ontology Language (OWL) is a W3C standardised way of representing such domain knowledge, and allows the representation of this domain knowledge as a subset of RDF. Using ontological inference, a large amount of logic usually programmed into specific applications can be united with the data store itself, leading to significantly simpler applications and greater interoperability. A fuller overview of ontologies in OWL and RDF is given by [16], and further description of the uses of ontology in this work is shown in section IV.

### III. AN ONTOLOGY-BASED ASSET MONITORING SYSTEM—AMAAS

#### A. Introduction to AMaaS

In late 2013, the University of Birmingham and Siemens Rail Automation responded to a funding call set by the UK Rail Safety and Standards Board on the topic of rail data integration—the ‘FuTRO Universal Data Challenge’. This challenge was set in order to further research into data integration as called for by the UK Rail Technical Strategy [2], and called on participants to develop novel approaches to combining data sources across the industry. One deliverable of the resulting project was AMaaS (Asset Monitoring As A Service), a software demonstrator showcasing the capabilities and advantages of using ontology and linked data in a railway asset monitoring application.

#### B. The AMaaS Platform

The AMaaS system described here was developed from an internal prototype asset monitoring platform. The original platform sought to re-think approaches taken by traditional rail asset monitoring systems [17] in order to provide a system that was easier to maintain, scale, and extend. Rather than using a centralised SCADA application to acquire and store data, AMaaS uses a cloud-based architecture based around a data store and a number of worker nodes, allowing data processing and storage resources to be provisioned only when they are needed. Field-based sensors and signal acquisition are placed near to monitored assets, and communicate with the AMaaS cloud-based core as necessary.

In addition to being scalable as data volumes increase and decrease, the implementation of a linked data store provides the ability to extend and develop the platform’s data model without altering existing applications. Domain ontologies allow application logic to be captured in the data model itself rather than in application code, greatly simplifying the entire system. The AMaaS project implements such a data model as an extension to the original application, providing a generic framework for asset monitoring that can be extended to encompass any number of more specific use cases, using the same application code.

#### C. Aims Of AMaaS Platform and Use Cases

The key aims of the AMaaS project, in line with the industry need described above, are as follows:

- Creation of a modular system that is scalable and extensible for data volume and variety. Using an RDF data store and OWL inference, instance data and domain knowledge are stored separately to applications, allowing new applications to be developed simply and quickly.
- Development of an application that information sharing between railway subsystems. Using linked data standards, it is possible to expose system data to other systems, and harness data from other systems, to prevent data silofication.
- Demonstration that a production-ready asset monitoring system can be built on standardised technology and off-the-shelf RDF toolsets. Commercial systems are usually built on widely-used and supported technologies, in order to reduce business risk associated with further industrial uptake.
- Demonstration of enrichment of asset data through data integration reasoning. Alignment of multiple data sources to OWL domain models allow ‘common sense’ scalable reasoning to infer facts that may be helpful. Examples of this include component composition, inverse relationships between entities, and subsumption.

The demonstrator was designed with two key use cases in mind, in order to show how data from disparate sources can be integrated and used to drive multiple applications. The first of these is to convey information about the state of

#### Table I

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>:Class390001 rdf:type</td>
<td>:Train</td>
<td>:TrainStation</td>
</tr>
<tr>
<td>:Class390001 :location</td>
<td>:CoventryStation</td>
<td></td>
</tr>
<tr>
<td>:CoventryStation rdf:type</td>
<td>:TrainStation</td>
<td></td>
</tr>
</tbody>
</table>
assets in the system for maintenance staff and asset managers, and is the primary use of an asset monitoring system. The second use case shows the resulting availability of railway infrastructure, based on asset states recorded from condition monitoring equipment and knowledge of infrastructure from other data sources. This type of information and presentation is likely to be useful to signalling and control staff as well as train operators, especially in times of degraded operation.

D. AMaaS Demonstrator Features

Reflecting the capabilities of a system driven using ontology and linked data, a list of features showing the demonstrator’s uses was created. Each stage involved is described in more detail in the following section, and shows one advantage or characteristic of using a linked data-based system.

1) **Points condition monitoring state.** In the first stage, the system uses its linked data store to present the user with information on several pieces of railway points monitoring equipment, including sensor characteristics, last known condition, and location. Statuses are viewed on a tabular web page, which queries the RDF store for information and links associated with each asset, and users navigate through the system as in a traditional web site. This stage duplicates existing functionality of other systems, and does not use any ontology inference.

2) **Diagnosis of fault severity.** From knowledge present in the ontology, the system infers ‘faulty’ or ‘healthy’ conditions on each observation, from the diagnoses given internally by AMaaS analysis tools (worker nodes). The AMaaS application then queries these observations against current time, and displays alarms where necessary.

3) **Mapping and inference of asset availability to infrastructure.** By creating links from the Siemens asset monitoring system to known infrastructure information (in this case, a track layout transcribed from hard copy track maps of Coventry Railway Station, UK), it is possible to infer the state of infrastructure based on asset monitoring observations. These are displayed in a signalling-style track layout, which users explore through clicking on components. By encoding dependencies between assets into the data store, inference is used to determine which parts of the system are available and unavailable based on other components’ state.

4) **Addition of new types of asset to the system.** To show the system’s extensibility, a Wheel Impact Load Detector device is added to the scenario. By extending the ‘asset monitor’ class in the ontology, the AMaaS system represents it correctly with no change to application code.

5) **Mapping of train schedule data into AMaaS data store, and Wheel Impact diagnosis.** Publicly available proprietary train schedule data [18] is mapped into RDF according to the ontology. As train location data is now linked to asset location, applications can be designed that query the interaction between the two. A ‘Train Finder’ view exploits this by allowing Wheel Impact Load Detector faults to be associated with rolling stock based on scheduled passing times.

E. System Architecture and Components

The AMaaS demonstrator system comprises not just of software systems, but of hardware data acquisition devices too. The principle components are as follows, and their composition shown in figure 2:
Applications

Fig. 2. Block Diagram Showing AMaaS System Architecture and Potential Applications

receives an RDF message analytics. In the demonstrator application, a worker node to be processed by a different worker nodes for storage and message queues to allow input data from a range of sources of its architecture given in figure 3. The AMaaS core uses computing techniques to allow scalability and extensibility. A and analysis of asset monitoring data, and uses cloud com-

Fig. 3. AMaaS Core Internal Architecture

- Asset monitoring and data acquisition hardware to obtain information about asset state (points machine current, track temperature).
- Bespoke aggregator devices that gather data, self-describe, and enter data into the AMaaS cloud.
- The AMaaS core, providing data storage and analysis functions, linked data endpoints and interfaces to other systems.
- The AMaaS web application, a bespoke web-based front end for railway asset monitoring.

The ontology and linked data store used in the AMaaS core are the focus of this paper, as well as the demonstrator front end provided by the web application.

F. The AMaaS Core

The AMaaS core provides the mechanism for processing and analysis of asset monitoring data, and uses cloud computing techniques to allow scalability and extensibility. A brief overview is provided below, and a simplified diagram of its architecture given in figure 3. The AMaaS core uses message queues to allow input data from a range of sources to be processed by a different worker nodes for storage and analytics. In the demonstrator application, a worker node receives an RDF message from one of a number of trackside aggregator devices or simulators, and enters the RDF data contained in this message into the application’s RDF triplestore (linked data store), as well as other data stores for fine-grained data if necessary. Additional worker nodes carry out further signal processing on some RDF messages, and add resulting knowledge into the triplestore.

G. Using Ontology Models in AMaaS

To provide a reference vocabulary and to enable useful inference, the AMaaS system is built on a railway core ontology developed at the University of Birmingham. This ontology draws upon industry standard vocabularies and data models, and defines key terms and relationships between them that are known to be true across the rail domain. Some of the assertions used in the ontology can be exploited by reasoning software, allowing the inference of new information given some real world fact. For example, if a Train X is known to stop at Place Y, then Place Y can be known to be a station, and thus an infrastructure entity. As this ontology is extended by adding more domain knowledge specific to the asset monitoring system, Place Y can also be inferred to be a 'Railway Asset', and treated as such in the application. In AMaaS, inference has three main uses:

- Flexible categorisation of entities. By using inference on subclass and superclass relations in an ontology, it is possible to infer which entities belong to certain classes. Thus, an application listing all asset monitoring devices can query the data store just for these, rather than having to specifically query for each type.
- Inference of fault information. By representing links between asset monitors and assets themselves, fault information can be associated with each piece of infrastructure using OWL DL reasoning. This allows other applications to use this data when displaying infrastructure information, and allows them to determine whether assets are available or unavailable.
- Inference of asset status, based on known dependencies. By creating links between assets that depend on each other’s functionality, availability of railway assets can be inferred based on the state of dependent assets. For example, if a points machine fails, the application infers that the piece of track adjacent to it is unavailable for use.

The ontology extensions built to facilitate these uses are discussed in the following section.

H. Implementation Technologies and Interoperability Standards

The whole of the AMaaS system was designed around W3C recommended standards for linked data usage. Such standards have enjoyed significant uptake in recent years, and tooling for working with semantic data models in RDF and OWL is now mature, with excellent commercial and open source offerings available for data storage, manipulation, and reasoning. The AMaaS Core and Web Applications were implemented in Microsoft C# .NET, and interfaces between system modules were easy to implement. Other technology choices were made as follows:

- The W3C SPARQL standard [19] provides a mature and capable way of querying and updating RDF data, and was used throughout the project via its standardised REST
API. SPARQL 1.1 is well supported by RDF toolsets, and was chosen for its ubiquity and ease of use.

- Clark & Parsia’s Stardog [20] RDF triplestore was chosen for AMaaS, due to its flexibility, documentation, and reasoning ability. Stardog’s web front end also allowed easy testing and diagnosis of problems during the project’s development.
- The dotNetRDF framework [21] allowed easy manipulation of RDF from within the .NET AMaaS core and web applications. dotNETRDF provides an internal graph model for data manipulation within C# .NET, and interfaces with Stardog via the SPARQL 1.1 Protocol.

The ability of an RDF triplestore to provide a standardised interface (endpoint) for accessing data from external applications is important when considering cross-industry interoperability. Currently, standards and tools for federated query across linked data sources exist, but the computation power required to service SPARQL requests means that responses from across different applications can be slow and unreliable. This is currently open problem in semantic web research, and solutions in this field are likely to be of use in enterprise interoperability applications such as AMaaS. One possible solution is in Linked Data Fragments [22], which significantly reduces the computation power required by linked data endpoints to service requests.

IV. Ontology Design Patterns and Implementation Details

A. Ontology Modules

In order to maintain extensibility into the future, and to aid compatibility with other systems, AMaaS is built upon a modular set of ontologies, which are imported and used as necessary. Upper level concepts are defined in an upper ontology, and a core rail domain ontology provides fundamental concepts and relationships likely to be useful in most applications. Three more specific models provide more detailed patterns and vocabulary for infrastructure, timetabling, and rolling stock data.

The AMaaS application ontology extends these core models to provide functionality for the AMaaS system, such as for sensors, observations, asset state, and proprietary metadata. Inference shown in the demonstrator application—detailed in figure 4—draws upon axioms present in many of these ontologies.

B. Upper Level Concepts, Rail Domain Fundamentals and Scalability

Upper level concepts such as time and space are represented in an upper ontology based on gold standard models in the literature. Where appropriate, best practice vocabularies are used to represent common concepts, including the following:

- Time concepts use the W3C standard Time Ontology, which provides ways of representing instants, intervals, and Allen time relations [23]. The AMaaS system takes a three dimensional approach to time representation, labeling entities where required with start times and end times according to this ontology. Whilst not ontologically ideal compared to a four dimensionalist approach as described in [24], this approach has the benefit of scalability and greater understandability.
- Position and location concepts are custom defined within the model, but include the W3C Geo Ontology for WGS84 GPS positioning. In AMaaS, Ordnance Survey positioning based on the OS Spatial Relations Ontology is also used [25].
- Units of measurement and measurement types are based upon the NASA QUDT ontology [26].
- Conceptual classes to categorise entities into types are based on ISO15926:2 [27]. The ontology classifies objects into independent (can exist in their own right), and dependent (existence depends on another entity, such as in the case of a measurement), which is useful in defining acceptable ranges and domains for properties.

The rail core vocabulary ontology is a result of work carried out manually constructing and curating knowledge from other domain models and from UK industry experts. The vocabulary and its sub-modules predominantly draw upon corresponding elements in RailML 2.2, relying on both its XML syntax and human-readable documentation in building an equivalent semantic data model.
C. Asset Monitoring Concepts, Patterns, and Outcomes

1) Representing Asset Monitoring Devices: To correctly communicate the semantics of the asset data stored within AMaaS, the ontologies used were extended with several new concepts to reflect the logical organisation of the Siemens asset monitoring system. A new ontology class EAssetMonitor was created that represents all possible asset monitoring devices, and is designed to be extended such that every new type of asset monitor used is a subclass. For points condition monitoring using the Siemens AMaaS hardware, the following subclasses were created:

- EI2MBox represents the class of ‘Aggregator’ devices—a data acquisition device with many sensors connected to it. One Aggregator may monitor several assets directly or indirectly.
- EVirtualMonitor is the class of logical asset monitors that each monitor one asset.
- EPointEndMonitor is a subclass of EVirtualMonitor specific to asset monitors that each monitor one point end. These are instantiated in the AMaaS demonstrator.

Asset monitor devices usually consist of many sensors. In AMaaS, they are connected through the :sensor relation, the semantics of which link one physical sensor to one asset monitor device. Each sensor is an instance of the :Sensor class or one of its subclasses, and each has its own characteristics (measurement unit, tolerance, etc.). :functionallyComprises and :physicallyComprises relations link asset monitors in the AMaaS system, such that the composition of the asset monitoring system is stored. These relations also allow reasoning to show the path to a fault on an asset monitor—if a virtual monitor observes a fault, its parent aggregator is also related to this fault through the :associatedObservation predicate. This pattern allows the asset monitoring system to function through an Entity Browser view, shown in figure 6. Relationships and inverse relationships link all parts of the asset monitoring hardware together, and thus a user can easily navigate to a particular device to inspect readings.

2) Observation Pattern and Storage: In order to store measurement events, an observation entity is created, linking together an asset monitor, a start time and a pointer to measurement data. In points monitoring, an observation entity is created on every points swing. Unlike other ontologies which encode discrete measurement values and units in RDF, the AMaaS system instead records measurement data, and stores fine-grained data in a separate data store optimised for speed and recall. As analysis is carried out by other parts of the system, diagnoses are asserted on these observation instances. The AMaaS application uses these instances to show the user entities which are at fault, and resolves fine-grained data where necessary from other systems.

3) Defining Healthy and Faulty Statuses: Through defining what diagnoses of observations are healthy and faulty, the AMaaS system can convey to applications the availability of railway assets and dependent infrastructure. Analysis algorithms in the AMaaS core assert a diagnosis on a condition observation using one of a number of possible diagnoses in the ontology, and from this the ontology can infer which assets are healthy and which are not. Asset availability is reflected in the web application’s Track View (figure 1), and in the Entity Browser previously mentioned.

Each possible diagnosis is created in the ontology as a subclass of :HealthyCondition or :FaultyCondition. Ontology reasoning uses this superclass to infer on the observation entity a type of :HealthyStateObservation or :FailedStateObservation, such that each application can simply query for members of these classes to obtain the state of each piece of monitoring equipment. Note that we do not infer Healthy or Failed states on the asset monitors themselves as each observation is time-dependent; this is a limitation of the pragmatic three dimensional approach taken to modeling time.

D. Infrastructure Linking and Dependencies

The next step in creating an interoperable asset monitoring system was to link information from the asset monitoring platform to infrastructure itself. In a traditional condition monitoring system, knowledge of the infrastructure associated with each piece of monitoring gear is either input manually at design time, for instance by designing a GUI around knowledge of the assets’ geography, or omitted entirely. However, using semantic data modeling allows the asset monitoring system to draw upon and integrate with infrastructure data from other sources; a demonstration of which is provided in AMaaS.

A section of track from the UK’s West Coast Main Line was modelled using the core rail ontology, with data taken from Network Rail Sectional Appendices [28] as required. This infrastructure data is visualised in the ‘Track View’ page in the AMaaS application (figure 1), and includes different routes, directional track, and crossings. The core ontology models track layouts as undirected graphs, with each node and arc as an entity itself in the model. As such, each infrastructure asset is uniquely identifiable, providing ways of creating links between asset monitoring equipment and the assets themselves. An extract from the infrastructure RDF code serialized into Turtle is shown in figure 7.

Links between the transcribed infrastructure information and asset monitoring equipment were created, through using a new relation, :monitors in the ontology. As such, each :VirtualMonitor entity monitors one points machine in the asset diagram, and inference is used to associate observations made by the asset monitoring system with the assets themselves through the :associatedObservation relation. As each observation also now has a ‘healthy’ or ‘failed’ status associated with it, the AMaaS web application simply queries for any failed status observations associated with an asset to show warnings to a user at a particular time—a query which continues to work regardless of the types of asset used.

To provide even greater functionality, the ontology can be exploited further. Dependencies between assets can be defined, such that if one asset fails, a dependent asset is also shown as unavailable. In AMaaS, a :dependsOn property
was created, and several assets were linked using this relation in the infrastructure data. Having added these assertions, the AMaaS track view correctly displays assets as unavailable once their dependent asset fails, without adjusting the track view’s application code in any way.

Whilst this is trivial to implement using application logic in a traditional system, it is the power provided when considering integration between systems that shows real benefit. When other systems utilise the linked data endpoint provided by AMaaS, data on asset health can be easily obtained and processed for uses other than maintenance—for example in signalling and route setting.

E. Rolling Stock and Timetable Information

1) Addition of Wheel Impact Load Detector: The final stage of the AMaaS demonstrator centres around the addition of data from a Wheel Impact Load Detector to the AMaaS system, and the mapping and import of passenger train schedule data from an external source. This enables computer-assisted resolution of wheel impact faults from the track-based WILD system to specific rolling stock by using ontology reasoning.

Wheel Impact Load Detectors (WILDs) are devices which are located on a piece of track, and detect wheel load and wheel impact of passing rolling stock. Systems used in the UK work by raising alarms with operators when detected values rise above set thresholds, but also rely upon maintenance staff with knowledge of train running times to manually match WILD information with corresponding vehicles. Ontology reasoning provides a simple way of minimising the effort needed to identify vehicles, and allow the inference of particular observations on the vehicles themselves, to align with rolling stock maintenance records.

In order to introduce WILD data into the AMaaS system, a new class was created in the ontology following the generic asset monitoring design pattern shown above. :EWILD was introduced as a subclass of :EAssetMonitor, meaning that the new WILD device data could be handled by the AMaaS application automatically. Observations generated by the WILD device are displayed both on the track view in AMaaS and as alarms (in the case of failed observations) with no change to application code, and measurement data is viewable through the Entity Browser.

2) Mapping of Train Schedule Data: Train schedule data was taken from a subset of publicly available ATOC CIF [29] files. After using an open source tool to convert these files from proprietary ASCII format into a MySQL database, mappings were defined from the database to the core ontology. Whilst tools exist for mapping from relational databases directly into RDF [30], OpenRefine [31] was used to create a subset of this schedule data and translate it into RDF. Identifiers for
station stops in schedule data were matched to identifiers in infrastructure data, and rolling stock and schedule data linked automatically.

Figure 8 shows a simplified version of the pattern used to represent calling points for railway services in the ontology. For each service imported, a number of :ServiceNode entities were created, each to signify one calling point for the service. Each node possesses properties for at least arrival and departure date/times and stop location; most nodes possess other properties too. It is the links between location, time and schedule that are of use in AMaaS, as they allow rolling stock faults to be calculated from WILD information.

3) Train Finder View: With schedule data and WILD device location data, AMaaS can infer which scheduled trains passed a WILD device at any time. The AMaaS Train Finder View exploits this to compile and display a list of locomotives that are likely to have caused an observed WILD fault. Upon viewing a WILD fault in the AMaaS system, a ‘Find Services’ function on the page queries the RDF data store for services passing that detector at that time, and presents the user with a list of likely matching services, shown in figure 9. Whilst this query would be possible without the aid of ontology reasoning, the use of inference allows the application query to be simplified greatly, and for the encoded domain knowledge to be stored in the data model rather than in the application.

Given the data available, it is possible to create software to automatically link WILD events with rolling stock, and it is even possible using ontology rule reasoning, which is not described here. However, the AMaaS platform does not aim to provide enforcement of complex business rules to run a whole system within the semantic data model itself; it is thought that the simple ‘common sense’ reasoning provided in the applications above provides enough benefit to foster interoperability between services, with little added complexity. Tools more suited to data analysis can easily be integrated with the system as worker nodes to provide non-monotonic reasoning and decision support for specific needs, such as in the case of points machine measurement analysis.

V. CONCLUSION (.5 PAGES)

A. Deliverables of AMaaS System

The AMaaS system described in this paper successfully demonstrates how semantic data models can be implemented to aid functionality and interoperability of an asset monitoring system designed for the rail industry. It shows that representation of data for such a system is easily achievable using RDF, and that through mapping and integration of linked data, new insights can be gained through reasoning over combined knowledge. Furthermore, the implementation shows how such systems can be built using well-supported, standardised technologies, and that other applications can take advantage of linked data stores through published linked data endpoints. Demonstration scenarios chosen for AMaaS provide examples of opportunities for such data integration between stakeholders, and highlight specific benefits that can be gained through integration of data sources.

B. Interoperability and Uptake

Whilst only one physical RDF data store was used in the AMaaS demonstration implementation, the abilities of the data model to act on data from disparate sources was shown. However, it is unlikely that all rail information systems will opt to use RDF data stores for business critical purposes—it is not a ‘magic bullet’ and is unsuitable in many situations. A more sensible approach to uptake of semantically interoperable systems may be the mapping of information into a publishable RDF form, similar to methods used by the linking open data movement [32], which encourages organisations to publish their publicly available data as RDF on the world wide web. Thus, an interoperability layer could be built atop legacy and non-RDF railway information systems that provides most of the benefits outlined earlier in this paper. Such a system would lower the barrier to implementation for smaller, non-essential information systems (where source data may already be available) and improve the capabilities of organisations in performing data analytics to provide new business insights into operations.

C. Limitations of AMaaS and RDF

Although benefits of RDF-based data models for rail applications have been shown, a number of limitations currently exist with the technology, which should be highlighted.

1) Performance and Scalability of Reasoning: In ontology reasoning, system designers must usually choose a compromise between the expressivity of their model and the performance they would like it to exhibit. For this reason, the second version of the Web Ontology Language (OWL2) [33] The ontology reasoning used in the AMaaS project is within the OWL 2 RL profile of the Web Ontology Language [33]. Whilst conforming to this profile reduces computation time significantly from OWL DL (where query answering is potentially undecidable), the OWL RL restrictions mean that in worst case query answering is NP-complete. In AMaaS, Stardog’s rule-based reasoning implementation showed good performance, with queries on the data store of 100000 triples.
always complete within two seconds on a laptop with Intel dual core i5 processor and 8GB RAM. As triple stores grow, reasoning scalability becomes an issue, and computation becomes costly. Given that across a rail information system may deal with huge amounts of data every day, this characteristic is relevant to production-ready RDF systems. Several possible solutions already exist however, and technologies such as streaming RDF and web-scale reasoning [34] provide new and exciting ways of overcoming the scalability problem. In the short term, design decisions and compromises can mitigate long computation times—either reasoning on small data subsets of large systems, or designing for predictable incomplete reasoning (where some facts are not inferred) allow performance savings.

2) Expressivity of RDF and OWL: Unlike conceptual ontological models, RDF and OWL place significant restrictions on how information can be represented, in order to maintain machine-readability an reasoning. The most obvious of these restrictions is the requirement for binary relations—a relationship can only ever be between two resources. The repercussions of this are that some concepts must be represented using verbose design patterns, some of which can cause triple counts in data stores to rise spectacularly [24]. As a result, system designers must choose between semantically proper representation of concepts and pragmatic shortcuts to allow most applications. AMaaS’ representation of time is not semantically correct, but is likely to suffice for the majority of applications built on it. Likewise, the AMaaS observation pattern is not as rich as those in other ontologies [35] but is expressive enough to encapsulate any use cases built on it.

3) Fine-grained Data: Due to the triple format, RDF is not good at representing very fine-grained data, such as each sample in a waveform for a points movement. As such, AMaaS uses another data store more suited for storing such data, and stores pointers to this store inside its RDF data model. In order to correctly represent one measurement using the AMaaS set of ontologies, nine triples are asserted. To store 15 seconds of data, sampled at 4kHz, for eight sensors, nearly half a million triples would be required. Even with optimisation of triplestores and slightly more conservative design patterns, this approach is inefficient, and unnecessary impedes reasoning times. The same problem is prevalent on the world wide web, and a new W3C working group has recently been set up to create a data format for use with linked data to solve this problem—CSV on the Web [36].

D. Further Development

Many of the limitations outlined above are the subject of ongoing research, as a result of the continued popularity of linked data on the web. None of these limitations prevent successful uptake of semantic data models in the rail industry, and many that were present in tools a few years ago have been solved by recent developments. Further projects between Network Rail and the University of Birmingham aim to lower the barrier to adoption of semantic data models for rail industry suppliers, and the continued investment by companies such as Oracle and IBM indicate that the technology will continue to mature.

VI. Acknowledgements

The author would like to thank Siemens Rail Automation, UK, as partners in the AMaaS project. The Universal Data Challenge was funded by Future Railways, and work on railway core ontologies was undertaken as a PhD CASE Studentship, funded by the Engineering and Physical Sciences Research Council.
REFERENCES


