A SEMANTIC APPROACH TO RAILWAY DATA INTEGRATION AND DECISION SUPPORT

by

RICHARD LEWIS

A thesis submitted to
The University of Birmingham
For the degree of
DOCTOR OF PHILOSOPHY

School of Electronic, Electrical
and Computer Engineering
The University of Birmingham
October 2012

ABSTRACT

The work presented in this thesis was motivated by the desire of the railway industry to capitalise on opportunities presented by new technology developments promising seamless integration of distributed data. This includes systems that generate, consume and transmit data for asset monitoring, maintenance and what is essentially decision support. The primary aim of the research was to investigate the limitations of previous syntactic data integration exercises within the domain, creating a foundation for state of the art semantic system development. The objective was to create some kind of modelling process that enables the domain experts to provide the domain structure concepts and semantic relationships between those concepts to the modeller. The purpose of the resulting model is to cater for the heterogeneity between systems supplying data that previous syntactic approaches failed to achieve and to integrate data from multiple systems such that the context of data is not lost when it is centralised for decision support applications.

The essence of this work is founded on two characteristics of distributed data management; the first is that current Web tools, such as XML, are not effective for all aspects of technical interoperability because it does not capture the context of the data; and second, there is little relationship between conventional database management systems and the data structures that are utilised in Web based data exchange which means that a different set of architecture components are required.

This thesis provides a background to syntactic and semantic data exchange and interoperability leading the reader through an ontology design process for semantic modelling relevant to the railway domain.

The specific outcome of the work are:

- A railway domain ontology – at the point in time when this this research was undertaken this outcome represented the first railway specific ontology resource aimed at information interchange.

- A railway rolling stock demonstrator application that utilised a network of semantic nodes and exchanging instance data related to the railway domain ontology. This application used a reasoner function to infer railway decision support information.

- A journal paper that addressed the limitation of conventional description logic models by extending them with probabilistic reasoning functions to capture tacit domain information.

The early chapters of this thesis review the wide ranging concepts related to knowledge management for decision support and introduce a solution to information management that addresses the challenges of technical interoperability. The latter chapters of this thesis describe industry led case studies that promote a generic approach to data exchange and integration.

It is shown through descriptions and examples how an ontology design can be constructed. The constituent parts of a design that support a Semantic approach are described in the

context of railway scenarios. The unique features that the solution brings are highlighted in each case giving the reader an insight into the differences between an ontology based approach and a solution based on a conventional database design.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my supervisor Professor Clive Roberts for his guidance and encouragement. I would also like to thank Rhys Davies for his assistance in deploying various aspects of the project prototype.

The research benefited significantly from the support of a number of industrial collaborators, in particular: Roger Shingler of Bombardier transportation, Gerhard Langer of Siemens Transportation, Florians Fuchs (formerly external research student at Siemens, Munich) and Michael Pirker of Siemens Research, Munich.

I would also like to thank my wife, Amanda for her understanding and help in affording me the time to complete this thesis.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
B2B	Business To Business
CPU	Central Processor Unit
DBMS	Data Base Management System
DCS	Distributed Control System
DL	Description Logic
DSS	Decision Support System
EWB	Engineers' Work Bench
HABD	Hot Axle Box Detection
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTP	Hyper Text Transfer Protocol
IP	Internet Protocol
KPI	Key Performance Indicator
OSACBM	Open System Architecture for Condition Based Maintenance
OWL	Ontology Web Language
RCM	Remote Condition Monitoring
RDF	Resource Description Framework
RMI	Remote Method Invocation
SCADA	Supervisory Control and Data Acquisition
SGML	Standard General Markup Language
SOAP	Simple Object Access Protocol
TCMS	Train Control and Management System
TCP	Transmission Control Protocol
VDU	Visual Display Unit
WILD	Wheel Impact Load Detection
WILM	Wheel Impact Load Measurement
WWW	World Wide Web
XML	eXtensible Markup Language
XSD	XML Schema Document

CHAPTER 1

INTRODUCTION

1.1 Background

In the railway domain, like many other large scale systems, there is drive to capitalise on opportunities presented by new technology developments promising seamless integration of distributed data. This includes systems that generate, consume and transmit data for asset monitoring and maintenance. Railway operators rely on technology to support the day to day running of the systems and as in other industries suffer from the same limitations. That is, all operator to system and system to system interactions need to be carefully specified and restricted. This is because the underlying interoperability is syntactic in nature and therefore designed to meet the specific needs of the application.

Previous work in the area of systems integration has addressed issues of standardisation for technical interoperability in the railway based on a fixed syntactic structure for the exchange of data and messages (Nash, et al 2004). A pilot study by Network Rail called Engineers' Work Bench investigated the implementation of current syntactic standards for information interchange. The work undertaken by Atkins Rail aimed to centralise data from all existing asset monitoring systems. The closeout report details that a fundamental limitation of the syntactic approach was the

expressivity to describe the data concepts. Manual configuration of the interfaces between the remote condition monitoring systems was required and each new system resulted in additional manual intervention. The result was a bespoke solution that was specific to the requirements of the remote systems. The syntax based XML schema that represented the interface to the legacy data systems grew with each new system and became effectively unmanageable as an interface specification.

Far from creating a generic solution the result was an interface that evolved with each additional monitoring system creating significant additional work effort. An additional problem was that the resulting repository neither matched the original data structure, nor was not strong enough to enable an efficient query and retrieval process (Elphick, 2003). The experience gained from this investigation mirrored those of other communities, where the syntactic representation of the data concepts was not expressive enough to capture the heterogeneity in the data.

Existing research by European railway stakeholders demonstrated that addressing interoperability issues by syntactic modelling had limitations. The conclusions were that centralisation of heterogeneous data between multiple stakeholders was not the answer as the interfaces were too rigid and the resulting data too difficult to interrogate once centralised (Shingler & Umiliacchi, 2003). The results achieved were not enough to gain a full understanding about diagnostic data, as there are limitations concerning the semantic aspects. Adding a semantic context by means of an ontology allows data to be turned into information which can be unambiguously understood and automatically elaborated by computers.

Significant potential value was placed on opportunities to generalise the interfaces and therefore the integration of IT systems in the European railways (Umiliacchi et al, 2007). Network Rail identified a need for an integrated approach to asset management and a change in maintenance philosophy stemming from opportunities from new technology implementations (Ollier, 2005). These case studies form an underlining motivation for the research because they demonstrate that there is high level of interest in solving the interoperability challenge.

Existing research results illustrate that syntactic based solutions did not fully realise the benefits of interoperability and therefore support the case for the investigation of a semantic approach to integration. Equivalent motivations for interoperability existed in the wider community where businesses wanted to exchange transaction and financial data and in industries that needed to centralise asset data. Information technology professionals were already addressing the challenges faced from the perspective of the data structure and centralisation (George, 2005). The centralisation of data became a key term in the work in this thesis because the early investigation covered current syntactic internet standards for data integration and centralisation whereas the resulting research addressed new standards for semantic data distribution. This distinction is important because centralising data creates huge repositories that require bespoke interfaces and significant maintenance, management and interrogation to create effective information for the users. Semantic data distribution on the other hand creates a lightweight mechanism to exchange information via generic interfaces where only the semantics of the data is centralised and shared. This approach creates significant potential value in the context of a multi-stakeholder, Pan-European railway system. The saving is in not needing to create and maintain large,

expensive repositories. This is because the stakeholders can maintain their existing data systems and only need to adapt an interface to ontology based systems that mirror the data that they are producing anyway. D2R is a tool that is already available to convert data from a conventional relational database into triples in a triple store matching a specified ontology design store data (Bizer and Cyganiak, 2006). This means that there is an opportunity to test the potential value of approach with relatively little associated hardware costs. The value that is more difficult to estimate is the potential for improved maintenance and increased efficiency in railway undertakings. This will require the stakeholders to deliberately take up and test the proposed technology. This research represents the first step in addressing this challenge.

As the demands on the railway system grow, so the drive to make effective use of system data through technical interoperability increases. This interoperability must be underpinned by standards that support a generic approach to data exchange and integration, such as those aimed at machine condition monitoring and diagnostics (ISO 13374-1, 2003) and industrial automation product data (ISO 10303-11, 1994). This was an underlining motivation for the research because the advancing communications and Internet technologies are broadening the scope and requirements for effective data interchange. There is now a demand for the development of standards for the interchange of railway data that support the implementation of the state of the art in computer architectures (Roberts, Lewis and Amoore, 2006).

The value of the proposed semantic approach is difficult to estimate in a railway system where stakeholders undertake various, numerous activities in a seemingly ad-

hoc manner. One major proposed cost benefit was the reduction of train delays due to more effective monitoring which is a major factor in the lifecycle costs of railway operation (Garcia Marquez et al, 2008). To gain an appreciation of the potential value of the proposed technology to the entire European Union the economic benefits to the member states need to be assessed. This subject is out of the scope of this research but it is important to note that the European Commission has made transportation, and specifically the interoperability of railway systems, a key specific area of research focus (EU Whitepaper, 2001). Demands to improve the performance of railway undertakings in the UK and throughout the rest of Europe led to the specification of the Strategic Rail Research Agenda (SRRA) for the European railway community (ERRAC, 2002). This agenda led railway rolling stock vendors and maintainers to review and extend the products and services they offered. An example of this change is an industry shift from using on board Train Control and Management System (TCMS) data and data from mandatory On Train Monitoring Recorders (OTMR) to support maintenance decision making (Prendergast, 2008). As a result of this trend, computer technologies associated with ubiquitous and pervasive information system architectures were under consideration for use in the railway domain. These technologies have particular relevance to condition monitoring and maintenance applications and are already implemented in some large scale industries, including oil and gas and power distribution (Campos, 2009), (Thurston, 2001), (McArthur et al, 2006).

Demonstrating the small scale benefits requires the analysis of localised life cycle costs where small improvements have significant potential to impact on the overall life cycle costs. The case studies described in this thesis aim at demonstrating some

increased efficiencies through the capture of tacit information in the semantics. The publication resulting from the work (provided in Appendix A) provides a practical demonstration of potential increased efficiencies of railway maintenance activities by capturing semantics in data and embedding it into the information system (Lewis & Roberts, 2010).

1.2 Scope of the Thesis

The motivations for the work presented in this thesis are the requirements of the railway industry to improve the way it uses existing legacy data. Legacy data resides in measurement systems related to various railway assets. The modern railway system has numerous sub-systems supporting applications ranging from train control (points and crossings) to power management and distribution (catenaries and substations). Numerous remote condition monitoring systems (RCM) have been developed to detect and diagnose the detrimental effects on these systems. For example, because of their potential to cause train delays, railway point machines, and systems for capturing incipient faults on them, have received considerable attention (Zhou et al, 2002). RCM systems also monitor vehicle health from the track side, on-board vehicle components, overhead catenary condition and other line side assets (Kesich and Golby, 2011), (Lehrasab et al, 2002), (Boffi et al, 2006), (Roberts et al, 2002). The objective of the research is to inform the development of a future railway IT infrastructure that will integrate data from numerous sub-systems and ultimately improve decision making activities.

The evolution of the railway system in the UK, which includes the privatisation and separation of the infrastructure and rolling stock parts, has resulted in sporadic investment and uncoordinated roll out of RCM systems. The lack of coherent planning across the industry has led to a situation where a significant investment has not led to an equivalent measureable benefit. This is caused, in part, by the lack of interoperability and standardisation in railway information systems, which limits the potential to aggregate RCM data and therefore restricts the timely analysis of events in the railway system (Stroud and Elphick, 2005).

Earlier attempts to address the issue of system interoperability and data integration were founded on a three tier architecture (3 tier architecture). The simplified diagram in Figure 1.1 illustrates how RCM data, collected by 'commercial off the shelf' (COTS) logging systems, is transported by Internet Protocol (IP) as eXtensible Markup Language (XML) data and stored in a data repository (data warehouse). The data is then accessible to an application layer where it is processed using analysis tools.

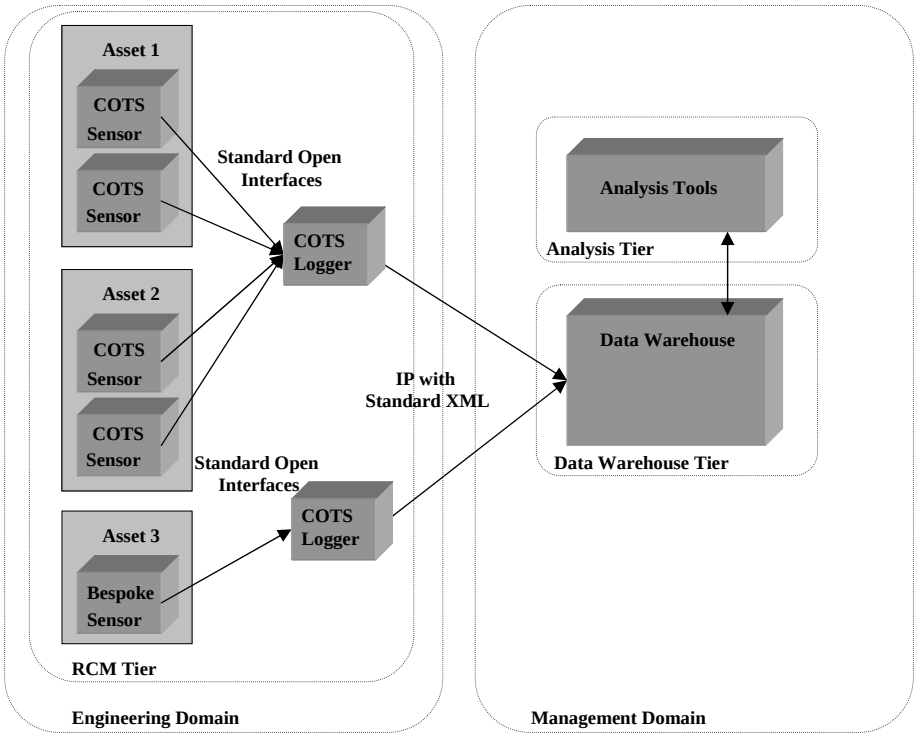


Figure 1.1 Three tier architecture

The investigation of the three tier architecture exposed railway analysts and developers to external groups working on standards aimed at asset data integration (Colins and Navathe, 2002). These groups were addressing the issue of interoperability and open exchange of data through the development of platforms for message exchange which implemented the available standards (Thurston and Lebold, 2001). The groups utilising these standards were concerned with domains that have particular characteristics, such as factories with plant and machinery. Assets housed at these facilities do not share the same characteristics with assets in the railways where:

- There is a requirement for associated asset management considerations such as component renewal and maintenance;
- There is a high geographical dispersion;
- There is a high degree of mobility;

- There are attributes which share contextual dependency with other, possibly external, influences.

Conversely, the integration and interoperability of data in the railway domain faces challenges that were not addressed by the current technology:

- The interpretation of an asset status is dependent on a number of contributing factors including component renewal and maintenance;
- Assets, and therefore associated IT systems, are highly geographically disparate;
- Rolling stock assets are mobile;
- IT systems, associated with similar types of assets, generate heterogeneous data;
- Legacy COTS type systems of various age and platform type represent a challenge to interoperability.

These challenges form the system level requirements for an interoperable IT system that supports data integration and informed decision making. Furthermore, from an operation perspective such an approach would need to allow:

- Data to be collected and stored without a clear view of its final use.
- Data to be collected and stored out of context of its usage, i.e. only a domain expert could interpret the data.
- Data of a particular type to be collected, such as that from an asset at one location/configuration that does not represent the same meaning as data from a similar asset at another location/configuration.

- Data to be structured and stored in a way that it can be readily operated upon by applications provided by multiple stakeholders.

Early trials did not cater for the evolving standards in RCM tier systems and initial groundwork resulted in developers posing a number of questions of the approach. The work presented in this thesis was motivated by the desire of the railway industry to address these issues and to answer the questions being posed, which include:

- What data should be collected?
  - The data collected is subject to application level implementation. However there are common features among many of the implementations e.g. all trains, regardless of class, design, etc have a number of common features. Therefore, the ontology design process addresses the capture of these common features through the development of design patterns. The data to be collected is then centred around capturing its impact on the context of related situations. A core ontology of concepts is then extended to meet the requirement of the particular application. The thesis addresses this requirement through the development of example design patterns which are extended with measurement concepts.
- How should contextual information about the conditions under which data was collected be conveyed to the analysts?
  - This matter is addressed in the design of the ontology where context is analysed and demonstrated through the description of a number of case studies. This thesis demonstrates how context is captured by deriving the implicit content to explicit content through reasoning. The

publication by the author and associated with this thesis demonstrates a case study for maintaining contextual information.

- How should the details of the different railway configuration and asset relationships at different locations across the network be captured?
  - The use of the descriptive nature of OWL DL ontologies presented in this thesis demonstrates how configurations and relationships between physical components can be represented. The creation of instance data related to the ontology illustrates how these concepts are represented.
- How can vast quantities of data be structured and stored in a way that enables generic computer algorithms to effectively translate it into meaningful information? (As opposed to relying on a suite of bespoke algorithms, targeted as specific data sets.)
  - A key objective of the work is to use the ontology as a common reference to access numerous resources. The key selling point of the research is that the data no longer needs to be centralised. The data can remain distributed and only the context and retrieval methods need to be shared.
- How does a national system accommodate the evolving standards in tier-one systems which will lead to variation in the quantity, quality and reliability of data provided about similar assets in different locations?
  - The evolution of the system is addressed through extensible nature of ontologies and version control. Requirements to interoperate with a new asset information system are met through the extension of an existing ontology model.

The subject of a semantic approach to railway data and integration and decision support is discussed in this thesis in the following manner:

Chapter 2 provides a review of concepts associated with decision support in the context of semantic technologies. This includes the definition of ontology concepts and the description of applications associated with the deployment of ontologies in data interchange and integration applications.

Chapter 3 introduces the idea of an ontology design process that is simple enough for rapid uptake yet sufficiently complex to support the retrieval of useful information and also the capture of context within it.

Chapter 4 describes the implementation of the ontology driven approach to address the challenge of semantic ambiguity experiences in previous syntactic approaches.

Chapter 5 considers a real railway integration scenario for describing the relationship between the railway vehicle and track. It demonstrates how an ontology based design can serve to overcome the heterogeneity in data by forming common concepts for different applications to relate to. It is shown how the ontology can capture some of the tacit knowledge of the expert and tie that knowledge to data through logic statements. This chapter also identifies a limitation of the ontology approach in dealing non-monotonic logic concepts.

Chapter 6 investigates the potential to apply probabilistic reasoning to the requirements of railway infrastructure monitoring.

Chapter 7 describes a small scale by providing a small scale demonstration of a real world application. The demonstration aims to illustrate the relationship between ontology modelling and a working application.

Chapter 8 presents the conclusions of the research along with recommendations on areas that require further work in order to allow technical interoperability to play its full role in improving railway system performance.

1.3 Information Exchange and Integration in Large Scale Systems

The discovery of requirements for electronic information storage and exchange can be attributed to the development of commercial mainframe computing in the mid 1960s. The success in large scale fabrication of semiconductor transistors led many industries to develop technologies that were previously impossible or impractical to deploy (Davis et al, 1964). For example, the smaller size transducers subsequently reduced the overall size of mainframe systems that previously inhibited their installation in a practical setting. IBM produced a family of mainframe CPU systems that enabled the exploitation of large data storage capacities, while providing flexibility in programming and interaction through the implementation of a number of peripherals. These systems provided Visual Display Unit (VDU) terminals that enabled direct programming and interrogation of the mainframe system. Even in these early stages, attempts were made to deal with issues of system design compatibility and legacy (Amdahl et al, 1964).

Industrial requirements for the control and condition monitoring of plant machinery led to conversion of analogue signals into the digital domain. These requirements subsequently led to the creation of the early Distributed Control Systems (DCS) (Samad et al, 2006). A DCS is a control system in which the controller elements are distributed throughout the system, rather than centralised like mainframes of earlier designs. Typically, these systems utilised proprietary interconnections and protocols which led to some limitation in integrating them with other commercial systems. The issue was resolved during the 1980s when the UNIX operating system and the associated networking technology, TCP/IP, were developed (Cardarella, 1990). This was an important step since the World Wide Web standard for messaging is founded on the TCP/IP protocol (Siever, 2003).

The architecture of information systems of this type remained largely unchanged until the creation of the Windows operating system. Windows offered the potential to create applications that could handle data from existing real time UNIX-based systems without the restrictions associated with real time characteristics – such as a un-friendly user interface and awkward programming environment. As a result a division was created between the 'real time' UNIX based control systems and the flexible 'application centric' Windows environment. An example of this type of system is a PC based Supervisory Control and Data Acquisition (SCADA) system (Wallace, 2003).

The telemetry standards for data exchange at the time enabled a remote PC based application to "dial-up" a network and upload data to it. However, this approach generally created a "standalone" structure as the system vendor typically provided a

proprietary interface for making contact, exchanging and processing data using that network. Another limitation of this approach was limited bandwidth of the modems used in the exchange of data.

PC driven TCP/IP communications, which underpins the Intranet, became ubiquitous as data driven activities became more widespread in industrial settings. Industries with large distributed assets, including the European railways, had invested in expensive telemetry based, standalone monitoring and control systems. The high cost of these existing systems meant that users were reluctant to update them. Any technology advancement that promised to integrate those legacy systems was therefore seen as valuable. For the first time, enterprise level solutions based on existing company Intranets were promoted to underpin knowledge management in decision support activities (Sulin et al, 1997).

The creation of the Internet and World Wide Web (WWW) technologies created opportunities to improve industrial data interchange. While the Hyper Text Transfer Protocol (HTTP) enabled the exchange of web pages between computers, the Hyper Text Markup Language (HTML) became the standard for encoding the contents of Web pages for transmission over HTTP. This meant that an application running on one PC could transmit a page containing the results of a process to a calling application. However, for industrial applications this approach was limited as the available communication bandwidth was not high enough for fast exchange of the data that is embedded in Web pages. The solution to this problem arrived as a subset of the Standard Generalised Markup Language (SGML), called the eXtensible Markup Language (XML) (Harold and Means, 2001). SGML was created as an

abstract meta-language for representing many concrete syntaxes[1], originally intended to enable sharing of machine readable documents (Clarke, 1997). The XML standard is a smaller derivative set intended to be much easier to process than the full set of SGML mark-up.

XML enables the exchange of data over HTTP without the overhead of HTML encoding i.e. a Web page is not required to support the exchange of data. XML shares a common meta-level feature of SGML called the Document Type Definition (DTD) but also presents a newer specification called an XML Schema Definition document (XSD) (Fallside, 2004). The XML schema provides a meta-level controlling document that is shared by applications and used to control the range and structure of data elements entered into the XML data document. The receiving application uses the schema document to determine the validity of data elements within the XML data. The implication of this development was the potential for true systems interoperability over a wide network, where devices and applications openly exchange data using the XML standard.

The term Business to Business, or B-2-B, became synonymous with the exchange of XML data between enterprise information systems. XML promised seamless integration of data at the business level creating the scope to perform multiple business transactions between systems that were previously incompatible. It offered platform independent data exchange that enabled systems built at different times, on different operating systems, running applications on different database management

[1] The syntax of a language including all the features visible in the source code such as parentheses and delimiters. The concrete syntax is used when parsing the program or other input, during which it is usually converted into some kind of abstract syntax tree (conforming to an abstract syntax).

systems and generating heterogeneous data (Yen et al, 2002). XML provides a single interchange format that supports these requirements, but one drawback is that different users can model data in different ways, which can lead to heterogeneities at different levels, including the semantic one (Cruz and Rajendran, 2003). Some very limited semantics can be drawn from the encoding of the sequence of elements and nesting in the schema, but this is too prohibitive to achieve real semantic interoperability. This means that the receiving application still requires commands to look for specific data items, which makes the data application specific. Arguably this situation represents only a small improvement on the SCADA based systems that were providing data integration over a decade earlier. In addition to these issues, since data representation in XML is not directly compatible with relational databases, some conversion is necessary.

Vendors of enterprise level integration systems attempted to overcome the heterogeneous nature of XML. For example, the Open Standard Architecture for Condition Based Maintenance (OSA/CBM) included some standardised schema for exchange of plant based condition monitoring data (ISO 10303-55:2005). It is this approach that lends itself to data integration and interoperability in large scale systems such as the railways.

1.4 Information Management

A challenge to achieving data integration in information management is the ability share information based on the intended meaning, i.e. the semantics of the data. The

database management system (DBMS) solution to mass data storage creates two barriers to effective integration:

- The ability to maintain the semantics of data is difficult because the context of data is lost during integration.
- There is a lack of extensibility; mapping techniques are required to map XML data to relational database tables. This approach is not 'extensible'; each additional system requires a new mapping and database configuration.

These barriers present important challenges to large scale systems integration where there are numerous systems producing heterogeneous data. The application of Semantic Web technology is proposed as a candidate solution for overcoming the limitations of XML schemas for business to business transactions (Trastour et al, 2003).

Work at Audi highlights the requirement to standardise the integration activity through the implementation of new Internet technology and methods (Antoniou and Van Harmelen, 2004). This situation is analogous to that of the National Health Service (NHS) where there are requirements to interoperate with numerous, multiple and heterogeneous information systems (Lewis, 1981) and other large scale systems with information management requirements such as power distribution and military applications (Feng et al, 2005), (Valente et al, 2005). Data heterogeneity can also arise from the federation of database systems resulting from one organisation becoming the owner of another. In such an instance the context of the data defined in each system becomes important during integration (Dey, 2001). The subject of context is an important characteristic in the field of pervasive computing where a

network of intelligent applications shares an *ontology*, enabling contextual information to be maintained during exchange and integration (Chen et al, 2004). The ontology maintains the semantics of the data exchanged between applications and provides the capability to share some common interpretation of that data.

The implementation of an ontology during the design and deployment of a system raises a number of characteristic challenges: first, the system stakeholders require a common understanding of why an ontology approach is required, what an ontology is and what it can represent; second, an agreement of a formal method for defining ontologies is required to enable a consistent approach for the developers and domain experts to adhere to; third, the way in which the ontology is represented needs to be addressed as this will have implications on the technology used to deploy the associated information system.

1.5 Knowledge Modelling

The objectives of an ontology design are related to the activity of collecting domain and application requirements. These requirements are based on the premise that existing approaches do not meet the needs of information system users because the detailed semantics of the data are not available. In general, the semantics are required for knowledge sharing and re-use across different applications. From an engineering perspective, the semantics are focussed on capturing the tacit knowledge of experts to support information sharing within a particular domain (Borst, 1997). This sharing is dependent on a vocabulary and, more importantly, on the conceptualisations that the terms in the vocabulary are intended to capture (Chandrasakaran et al, 1999).

Ontologies are an essential ingredient in capturing the conceptualisations that underpin knowledge; without such conceptualisations there cannot be a vocabulary for representing knowledge. However, the exact definition of an ontology is open to interpretation and is dependent on the domain within which it will be used and the specific requirements for conceptualisation. Guarino and Giaretta provide a discussion of a number of interpretations of the term ontology (Guarino & Giaretta, 1995). Noy and & McGuinness define an ontology as ‘‘a formal explicit description of concepts in a domain of discourse” (Noy & McGuinness, 2001).

1.6 Contribution of the Thesis

The development of a semantic approach to data integration demands the consideration of a number of technical facets. These include an understanding of the IT systems and architectural components that underpin the deployment; the process of capturing the ontology to meet the requirements of the applications and the users, typically involving all of the stakeholder; the capture, formatting and storage of the defined ontology patterns and the resulting ontology concepts; the deployment of triple stores, reasoners and frameworks to test the implementations; the population of repositories with sample data supporting testing and demonstration and the development of the applications to demonstrate the integration and decision support capability of the resulting process. Furthermore, significant effort was made in gaining the opinions and contributions of industry stakeholders who held the knowledge of the systems and the problems faced by the sector. The novel contributions resulting from these activities cover:

- the development of a number of ontology design patterns supporting the foundation of a simplified ontology design process.
- the development of a Railway Domain Ontology (RDO) that forms a core ontology that captures the semantics of railway concepts and can be imported and extended by external stakeholders.
- the description of a hierarchy of interoperating reasoning nodes that utilise the RDO as part of a common interface specification.
- the development of extensional ontologies containing concepts that support the demonstration of practical railway applications.
- the introduction of description logic embedded in modern ontology standards and used in the context of industrial data management.
- the demonstration of the capture and utilisation of tacit, context information in a reasoner application as a stakeholder demonstration - a publication describing the advancement of ontology design to probabilistic reasoning with practical application from the railway domain was created.

1.7 Conclusions

This chapter discusses opportunities to implement modern IT solutions that incorporate aspects of pervasiveness and ubiquity to within the railway domain. It explains the evolution of network computing and highlights the importance of addressing the challenge of system compatibility and legacy system integration in large scale systems. The chapter also demonstrates how the current state of the art in the railway domain has led to the creation of application specific data models during

system design and describes how this has resulted in a limited return on investment in effective decision support mechanisms.

The issue of data context is a key target for the railway domain as effective decision support is dependent upon it. The information system architecture that is deployed needs to deal with the integration of many types of information systems, as well as the evolution of those systems. In this work, the context in railway domain data integration is captured in a schema model based on ontology methods. The extensibility of railway domain information systems is demonstrated through the modelling techniques available to the ontology design process.

The work undertaken covers the analysis of ontologies for domain data integration, focussing on four areas of research:

- Technology concepts enabling semantic integration for decision support
- Ontology development techniques for domain data integration
- Ontology data modelling
- Application layers including prototype deployment

By addressing the state of the art in this subject area, there is an opportunity to provide a novel solution to heterogeneous data integration and system interoperability for decision support within the railway domain.

CHAPTER 2

REVIEW OF KNOWLEDGE MANAGEMENT FOR DECISION SUPPORT

2.1 Introduction

This chapter provides a review of concepts associated with decision support in the context of semantic technologies. This includes the definition of ontology concepts and the description of applications associated with the deployment of ontologies in data interchange and integration applications.

The Semantic Web represents a growing set of standards, technologies and applications that build on existing resources defined as the World Wide Web as we currently know it. The World Wide Web changed the way we communicate, the way we do business and the way we seek information and entertainment. Calling it the next step in Web evolution, Tim Berners-Lee, the inventor of the World Wide Web, defines the Semantic Web as "a web of data that can be processed directly and indirectly by machines". The semantic theory provides an account of "meaning" in which the logical connection of terms establishes interoperability between systems (Shadbolt et al, 2006).

Considering the current Web, data is stored in the form of documents on which computer algorithms perform key word searches. However, the computer can only present the data, it cannot understand the data well enough to display the content most relevant to a given context. The Semantic Web consists of a combination of data as well as documents on the Web so that machines can process, transform, assemble, and even act on that data in useful ways.

Technologies created and under development in the Semantic Web show potential for solutions in industrial data interchange and integration applications such as decision support. The following sections aim to describe the technology in the context of large scale systems integration and the decision support challenges defined in the previous chapter.

2.2 Knowledge Management Concepts

In order to respond to new challenges in an increasingly complex and dynamic world, modern management uses a vast amount of knowledge from various sources. Advances in organizational and network computing technologies and developments such as corporate digital libraries and data warehousing technology have helped to address the challenges faced by large corporations. These technologies were appended with computer supported decision making and problem solving environments (Ba et al, 1997). Research in the decision sciences has resulted in the development of a variety of Decision Support Systems (DSS) that are useful in solving many decision problems faced by individuals and organisations. It is now possible to access these DSS using the Internet. However, in spite of the variety of ways in which information is discovered on the Web, it is still difficult to find many

of the resources that are available and to assess the quality of the information that is found. *Metadata* is one mechanism implemented to facilitate both the location of specific Web content and the assessment of its quality (Gregg et al, 2002). The elaboration of such systems has been applied to applications such as health care administration (Pederson and Larson, 2001) and the planning of maintenance in the railway domain (Dell'orco, 2003).

In combination the Internet, the Web, and telecommunications technology can be expected to result in organizational environments that will be increasingly more global, complex, and connected. Supply chains will be integrated from raw materials to end consumers, and may be expected to span the planet. Organizations will interact with diverse cultural, political, social, economic and ecological environments. The importance of knowledge as an organizational asset in these interactions explains the increasing interest of organizations in knowledge management. Since organizational knowledge is derived from individual knowledge; Knowledge Management Systems (KMS) must support the acquisition, organization and communication of both tacit and explicit knowledge of employees (Rubenstein-Montano et al, 2001).

According to Shim et al, making tools like neural networks, decision trees, rule induction, and data visualization widely available to naive users using Web technologies would be a mistake (Shim et al, 2002). Therefore the responsibility for adoption of tools to support knowledge management lies with the organisation. However, few organizations have a rigorous, comprehensive knowledge management methodology for creating knowledge management systems and initiatives. Part of the reason for this phenomenon is that the knowledge management providers themselves

do not have comprehensive knowledge management methodologies. This situation can lead to confusion and scepticism in the marketplace created by overhype and mislabelling of tools. This happened in the past to the artificial intelligence (AI) community, which flowed through peaks and valleys until the expectations were brought into alignment with capability (Leibowitz, 2001). A similar assertion is made for knowledge management; the expectation of the creation of computer content that represents knowledge needs to be matched with IT efforts and changes in organisation strategies that support the delivery of that expectation (Bloodgood and Salisbury, 2001).

The Advent of Web standards and tools that are aligned with the ambition to turn the World Wide Web into a "semantic" system rather than a "syntactic" system[1] presents an opportunity for businesses to adapt the methodology for data capture, storage and processing to create knowledge. The difference between syntactic and semantic interoperability underpins the work in this thesis. The work was initiated through discussion with a number of stakeholders who had experienced the challenges of syntactic integration over the Web. Fundamentally, Syntactic interoperability relies on specified data formats, communication protocols, and the like to ensure communication and data exchange. The systems involved can process the exchanged information, but there is no guarantee that the interpretation is the same. Semantic interoperability, on the other hand, exists when two systems have the ability to automatically interpret exchanged information meaningfully and accurately in order to produce useful results via deference to a common information exchange reference

[1] This refers to a system that operates by attributing meaning to terms rather simple key word searching

model. The content of the information exchange requests are unambiguously defined: what is sent is the same as what is understood (Ide and Pustejovsky, 2010).

Syntactic interoperability relies on a process of conceptual modelling which results in a schema that is used to verify the resulting data (Dunn and Gabski, 2001). The focus is on the validity of communicated data, rather than any implied meaning, and metadata is limited to a conceptual schema which is used to verify the content. A semantic system consists of various levels of metadata that represent the annotation of the instances of data concepts and the relationships between terminology concepts in the domain. The focus is on the interpretation of the meaning of any instance of a concept from its annotation and relationship to other concepts in the hierarchy (Berners-Lee, 2001).

The computer science community has adopted the term "ontology" as a generalised reference to the conceptualisations that underlie knowledge. An Ontology can represent a conceptualisation of ideas and relationships between computerised terminology and can therefore represent the meaning of the terms in a particular domain. In this context, this meaning is referred to as semantics, from which the term Semantic Web was derived. The idea of Ontology, in particular the adoption of Semantic Web tools, is proposed as a solution to the information and knowledge management and decision support needs of modern organisations.

2.3 Ontology Definition

The term *semantics* refers to the study of meaning and the interpretation of that meaning in a given context. In the context of information management, this term has become more relevant as the amount of data available to us has increased exponentially. This is because the technology that produces and consumes this data limits the ability to interpret it as information if it cannot interpret the semantic information available. Semantic technologies are aimed at providing more explicit meaning for the information that is at our disposal and therefore improving the interpretation of data into information.

Supporting the interpretation of data into information requires the capture of the semantics of that data. One method of capturing those semantics is to create a model of concepts, referred to as an ontology. The term *ontology* is rooted in early theology and a branch of philosophy known as metaphysics. Students of Aristotle first used the word 'metaphysica' (literally "after the physical") to refer to what their teacher described as "the science of being *qua* being" - later known as ontology. The Oxford English Dictionary interpretation of the term ontology is "*the science or study of being; that branch of metaphysics concerned with the nature or essence of being or existence.*" Computer science has extended the meaning of the term to suit its own purpose. However, there is still some variation in the exact meaning; according to Guarino "*an ontology refers to an engineering artefact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words. This set of assumptions has usually the form of a first-order logical theory*" (Guarino, 1997);

In the context of railway data interoperability and more specifically the domain examples presented later in this thesis, the adopted definition is defined as: *"Ontology is similar to a dictionary, taxonomy or glossary, but with structure and formalism that enables computers to process its content. It consists of a set of concepts, axioms, and relations, and represents an area of knowledge. Unlike taxonomy or glossary, ontology allows the modelling of arbitrary relations among concepts while also capturing logical properties and semantics of the relations such as symmetry, transitivity and inverse, and logically reason about the relations"* (Lee and Goodwin, 2006).

2.4 Description Logics

The sections presented so far have introduced concepts of information management. What is necessary to meet the objectives of a semantic approach is means of representing knowledge and reasoning with that knowledge. Knowledge representation and reasoning is a subject of Artificial Intelligence (AI). Therefore, the AI concepts that are necessary to underpin the development of a semantic based system are introduced in the context of a practical implementation for railway decision support.

Description Logics evolved out of difficulties in overcoming representation and semantic difficulties in "semantic network" and frame languages (Quillan, 1968), (Minsky, 1981). This foundational work led to the differentiation between "terminological" information, which captured definitions and relations among concepts, and "assertional" information, which captured information about individuals

in the domain (Brachman and Levesque, 1982). This is an important distinction because it represents the difference between the structure of the domain knowledge and the actual conditions at a particular point in time. In the work described in this thesis the aim was to capture the way in which information could be stored so that it can be used at different times to support decisions in various scenarios. This early work formed the foundation for knowledge representation and later underpinned the development of early reasoning systems (Brachman et al, 1985). The importance of expressivity was discovered during these early exercises (Patel-Schneider et al, 1984). This is important to the proposed research because data added to the system must enable an appropriate assessment to be made. In support of this a means of semantic control is required within the information model. The challenge of restricting the semantics to enable decidability was still to be addressed.

In order to record information as concepts that are interrelated and have some meaning, it is first necessary to have a formal means of representing that information and then reasoning over it. First Order Logic (FOL) provides a formal language for representing knowledge (Israel, 1983). FOL consists of a syntax made up of logical and non-logical terms that enables the capture of the semantics of the domain. Logical symbols consist of three types:

> *Punctuation:* "(," ")," "," and "."
>
> *Connectives:* "∀" means "for all", "∃"means "there exists", "¬" logical negation , "^" logical conjunction, " " logical disjunction, "=" logical equality.
>
> *Variables*: x, y, z

Non – logical symbols are those that have an application dependent meaning or use. There are two types of non-logical symbols:

Function symbols: bestFriend

Predicate Symbols: OlderThan

In FOL these symbols are used to create expressions in the form of *terms* and *formulas* that represent knowledge of the domain. A term is used to refer to something in the world and a formula used to express a proposition (Gabbay et al, 1998).

In a logic based approach, the representation language is usually a variant of first order predicate calculus. FOL provides a powerful and general machinery for reasoning, where reasoning amounts to verifying logical consequences. However, FOL does suffer from limitations in that it does not cater for more complex constructions such as "a man whose children are all female". Expressing knowledge as complex predicates was made possible in network based systems developed in the early stages of knowledge modelling. However these early solutions suffered from a lack of precise semantic characterisation which resulted in similar systems behaving very differently. An important step in the evolution of semantic systems was the realisation of frames which could be given a semantics by relying on first order logic (Hayes, 1979). This development was the precursor to the research in the field of Description Logics.

Description logics (DLs) are a family of knowledge representation languages used to represent the knowledge of an application domain in a structured and formally well-understood way. The name is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions and on the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics (Baader et al,

2003). This is important in the railway domain where there is a need to produce knowledge models that meet specific role requirements. The focus needs to be on the information rather than the development process and therefore a formal semantics is a key ingredient. This is why DLs were considered within the proposed research.

The foundations of description logics are concept and role descriptions (or concepts and roles for short). Intuitively, a concept represents a class of objects sharing some common characteristics, while a role represents a binary relationship between objects, or between objects and data values. These role relationships are particularly important in domains which are composed with physically composed objects (Padgham and Lambrix, 1994). This was specifically relevant to the railway domain where the requirement was to integrate heterogeneous data from various distributed and mobile assets. The capability to capture the physical composition was believed to be an essential and valuable aspect of the technical development.

FOL shares some similarities to DL in that it has two symbols - logic symbols, which have a fixed meaning or use and non-logical symbols, which are application dependent. However the symbols used in DL from FOL as there are more logical symbols and the specification of non-logical symbols is different. The logical symbols of DL are:

1. *punctuation:* "[," "]," "(," ")"
2. *positive integers:* 1, 2, 3, etc
3. *concept forming operators:* "**ALL**," "**EXISTS**,""**FILLS**," "**AND**"
4. *connectives:* "→"

There are three types of non-logical operators:

1. *atomic concepts:* Person, FatherOfOnlyGirls
2. *roles: :Child, :Height. :Employer*
3. *constants: desk13, carNo51*

In DL there are two additional syntactic expressions: *concepts* and *sentences*. Concepts are combinations of concept forming operators e.g. if *r* is role and *d* is concept, then {**ALL** *r d*} is concept. A sentence is a collection of concepts that are related by some logical connective. Knowledge formulated in DL by a collection of sentences in a *knowledge base*. The work of Rector provides practical examples that support the formulation of these sentences (Rector et al, 2004). These logic representation characteristics are embedded as annotations of the ontology representation language. The important feature of these annotations is that they are machine interpretable, transforming them into semantic annotations (Horrocks, 2008). This is an essential characteristic of a semantic based system.

There are a number of types of DLs which dictate the expressivity and decidability of the resulting ontology species. The language of $\mathcal{AL}$ (acronym for 'attributive language') was introduced as a minimal language that is of practical interest (Schmidt Schaub and Smolka 1991). The other languages of this family are extensions of $\mathcal{AL}$. Each additional constructor is associated with a specific capitol letter. In modern DLs the language *S* is often used as a minimal language, which has been previously called $\mathcal{ALC}_{R+}$ according to the above convention. This language is referred to by the symbol $\mathcal{S}$ because it relates to the propositional multi-model logic S4 (Schild, 1991), this approach avoids names becoming too cumbersome when adding letters to represent

additional features (Horrocks et al. 2000). Well known S family languages include $\mathcal{SI}$ (Horrocks et al.1998), $\mathcal{SH}$, $\mathcal{SHI}$ (Horrocks and Sattler, 1999), $\mathcal{SHf}$ (Horrocks, 1997), $\mathcal{SHIQ}$ (Horrocks et al, 2000), $\mathcal{SHOQ}$(D) (Horrocks and Sattler, 2001), etc.

The various flavours of OWL ontology are based on these languages. For example, OWL Lite is based on SHIF and OWL DL is based on $\mathcal{SHOIN}$ (Horrocks et al, 2007). $\mathcal{SHOIN}$ provides most expressive means that one could reasonably expect from the description-logical basis of an ontology language, and was designed to constitute a good compromise between expressive power and computational complexity/practicability of reasoning. It lacks, however, e.g. qualified number restrictions which are present in the DL considered here since they are required in various applications (Wolstencroft et al., 2005) and do not pose problems (Horrocks & Sattler, 2007).

The basic languages are represented by:

$\mathcal{AL}$ Attributive language. and

$\mathcal{FL}$ Frame based description language

Which are then followed by a number of extensions from:

$\mathcal{F}$ Functional properties.

$\mathcal{E}$ Full existential qualification (Existential restrictions that have fillers other than `owl:Thing`).

$\mathcal{U}$ Concept union.

$\mathcal{C}$ Complex concept negation.

$\mathcal{H}$ Role hierarchy (subproperties - rdfs:subPropertyOf).

$\mathcal{R}$ Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.

$\mathcal{O}$ Nominals. (Enumerated classes of object value restrictions - `owl:oneOf`, `owl:hasValue`).

$\mathcal{I}$ Inverse properties.

$\mathcal{N}$ Cardinality restrictions (`owl:cardinality`, `owl:maxCardinality`).

$\mathcal{Q}$ Qualified cardinality restrictions (available in OWL 2, cardinality restrictions that have fillers other than `owl:Thing`).

$(\mathcal{D})$ Use of datatype properties, data values or data types.

$\mathcal{S}$ Is and exception which forms an abbreviation for $\mathcal{ALC}$ with transitive roles.

$\mathcal{ALC}$ is simply $\mathcal{AL}$ with complement of any concept allowed, not just atomic concepts. This forms the foundation of the description logic $\mathcal{SHIQ}$ which is the logic $\mathcal{ALC}$ plus extended cardinality restrictions, and transitive and inverse roles. $\mathcal{SHOIN}$(D) is DL with logic $\mathcal{ALC}$ plus extended cardinality restrictions, transitive and inverse roles, plus enumerated classes and use of datatype properties and role hierarchy (Horrocks et al, 2003).

Since OWL DL makes use of a $\mathcal{SHOIN}$ (D) DL the expressivity in any reasoning is bound by this specification. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the meta-modelling facilities of RDF Schema (e.g. defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable since complete OWL Full implementations do not currently exist. The important issue in

the context of practical information interchange is that OWL is syntactically embedded into RDF, and therefore all of the RDF serializations can be used. RDF/XML is the normative syntax for information interchange (Orbitko and Smid, 2004).

As discussed, OWL DL is the description logic $\mathcal{SHOIN}$ with support of data values, data types and datatype properties, i.e., $\mathcal{SHOIN}$(D), but since OWL is based on RDF(S), the terminology slightly differs. A concept from DL is referred to as a class in OWL and a role from DL is referred to as a property in OWL. This provides a subtle but important variation between the two specifications. Terms in an OWL-DL ontology have a specific semantics in the logic representation of the DL language as shown in table 2.1 below. Further, the terms in Table 2.1 form part of the expressions of the axioms and facts derived to represent the knowledge of the domain. Table 2.2 provides a summary of the expressions formed as axioms and facts in OWL-DL.

Table 2.1 OWL DL descriptions, data ranges, properties, individuals and data values syntax and semantics

Abstract Syntax	DL Syntax	Semantics
Descriptions (C)		
A (URI Reference)	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
owl:Thing	$\top$	$\mathtt{owl:Thing}^{\mathcal{I}} = \Delta^{\mathcal{I}}$
owl:Nothing	$\bot$	$\mathtt{owl:Nothing}^{\mathcal{I}} = \emptyset$
intersectionOf(C_1 C_2 ...)	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
unionOf(C_1 C_2 ...)	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
complementOf(C)	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
oneOf(o_1 ...)	$\{o_1, ...\}$	$\{o_1^{\mathcal{I}}, ...\}$
restriction(R someValuesFrom(C))	$\exists R.C$	$\{x \mid \exists y.\ \langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
restriction(R allValuesFrom(C))	$\forall R.C$	$\{x \mid \forall y.\ \langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
restriction(R hasValue(o))	$R:o$	$\{x \mid \langle x, o^{\mathcal{I}}\rangle \in R^{\mathcal{I}}\}$
restriction(R minCardinality(n))	$\geqslant nR$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid \langle a,b\rangle \in R^{\mathcal{I}}\} \geq n\}$
restriction(R maxCardinality(n))	$\leqslant nR$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid \langle a,b\rangle \in R^{\mathcal{I}}\} \leq n\}$
restriction(U someValuesFrom(D))	$\exists U.D$	$\{x \mid \exists y.\ \langle x,y\rangle \in U^{\mathcal{I}} \wedge y \in D^{\mathcal{D}}\}$
restriction(U allValuesFrom(D))	$\forall U.D$	$\{x \mid \forall y.\ \langle x,y\rangle \in U^{\mathcal{I}} \rightarrow y \in D^{\mathcal{D}}\}$
restriction(U hasValue(v))	$U:v$	$\{x \mid \langle x, v^{\mathcal{D}}\rangle \in U^{\mathcal{I}}\}$
restriction(U minCardinality(n))	$\geqslant nU$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid \langle a,b\rangle \in U^{\mathcal{I}}\} \geq n\}$
restriction(U maxCardinality(n))	$\leqslant nU$	$\{a \in \Delta^{\mathcal{I}} \mid \#\{b \mid \langle a,b\rangle \in U^{\mathcal{I}}\} \leq n\}$
Data Ranges (D)		
D (URI reference)	D	$D^{\mathcal{D}} \subseteq \Delta^{\mathcal{D}}_{\mathcal{D}}$
oneOf(v_1 ...)	$\{v_1 ...\}$	$\{v_1^{\mathcal{D}} ...\}$
Object Properties (R)		
R (URI reference)	R	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
	R^-	$(R^{\mathcal{I}})^-$
Datatype Properties (U)		
U (URI reference)	U	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{D}}_{\mathcal{D}}$
Individuals (o)		
o (URI reference)	o	$o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$
Data Values (v)		
v (RDF literal)	v	$v^{\mathcal{D}}$

Table 2.2 OWL DL axioms and facts

Abstract Syntax	DL Syntax	Semantics
Classes		
Class(A partial C_1 ... C_n)	$A \sqsubseteq C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} \subseteq C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
Class(A complete C_1 ... C_n)	$A \equiv C_1 \sqcap ... \sqcap C_n$	$A^{\mathcal{I}} = C_1^{\mathcal{I}} \cap ... \cap C_n^{\mathcal{I}}$
EnumeratedClass(A o_1 ... o_n)	$A \equiv \{o_1, ..., o_n\}$	$A^{\mathcal{I}} = \{o_1^{\mathcal{I}}, ..., o_n^{\mathcal{I}}\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$	$C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$
EquivalentClasses(C_1 ... C_n)	$C_1 \equiv ... \equiv C_n$	$C_1^{\mathcal{I}} = ... = C_n^{\mathcal{I}}$
DisjointClasses(C_1 ... C_n)	$C_i \sqcap C_j = \bot, i \neq j$	$C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset, i \neq j$
Datatype(D)		$D^{\mathcal{D}} \subseteq \Delta^{\mathcal{D}}_{\mathcal{D}}$
Datatype Properties		
DatatypeProperty(		
U super(U_1) ... super(U_n)	$U \sqsubseteq U_i$	$U^{\mathcal{I}} \subseteq U_i^{\mathcal{I}}$
domain(C_1) ... domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$	$U^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{D}}_{\mathcal{D}}$
range(D_1) ... range(D_l)	$\top \sqsubseteq \forall U.D_i$	$U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times D_i^{\mathcal{D}}$
[Functional])	$\top \sqsubseteq \leqslant 1\, U$	U_i is functional
SubPropertyOf(U_1 U_2)	$U_i \sqsubseteq U_2$	$U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
EquivalentProperties(U_1 ... U_n)	$U_1 \equiv ... \equiv U_n$	$U_1^{\mathcal{I}} = ... = U_n^{\mathcal{I}}$
Object Properties		
ObjectProperty(		
R super(R_1) ... super(R_n)	$R \sqsubseteq R_i$	$R^{\mathcal{I}} \subseteq R_i^{\mathcal{I}}$
domain(C_1) ... domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$	$R^{\mathcal{I}} \subseteq C_i^{\mathcal{I}} \times \Delta^{\mathcal{D}}_{\mathcal{D}}$
range(C_1) ... range(C_l)	$\top \sqsubseteq \forall R.C_i$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times C_i^{\mathcal{I}}$
[inverseOf(R_0)]	$R \equiv (R_0^-)$	$R^{\mathcal{I}} = (R_0^{\mathcal{I}})^-$
[Symmetric]	$R \equiv (R^-)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^-$
[Functional]	$\top \sqsubseteq \leqslant 1\, R$	$R^{\mathcal{I}}$ is functional
[inverseFunctional]	$\top \sqsubseteq \leqslant 1\, R^-$	$(R^{\mathcal{I}})^-$ is functional
[Transitive])	$Tr(R)$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$
SubPropertyOf(R_1 R_2)	$R_i \sqsubseteq R_2$	$R_1^{\mathcal{I}} \subseteq R_2^{\mathcal{I}}$
EquivalentProperties(R_1 ... R_n)	$R_1 \equiv ... \equiv R_n$	$R_1^{\mathcal{I}} = ... = R_n^{\mathcal{I}}$
Annotation		
AnnotationProperty(S)		
Individuals		
Individual(		
o type(C_1) ... type(C_n)	$o \in C_i$	$o^{\mathcal{I}} \in C_i^{\mathcal{I}}$
value(R_1 o_1) ... value(R_n o_n)	$\{o, o_i\} \in R_i$	$\{o^{\mathcal{I}}, o_i^{\mathcal{I}}\} \in R_i^{\mathcal{I}}$
value(U_1 v_1) ... value(U_n v_n))	$\{o, v_i\} \in U_i$	$\{o^{\mathcal{I}}, v_i^{\mathcal{I}}\} \in U_i^{\mathcal{I}}$
SameIndividual(o_1 ... o_n)	$o_1 = ... = o_n$	$o_1^{\mathcal{I}} = ... = o_n^{\mathcal{I}}$
DifferentIndividuals(o_1 ... o_n)	$o_i \neq o_j, i \neq j$	$o_i^{\mathcal{I}} \neq o_j^{\mathcal{I}}, i \neq j$

These tables represent the expressivity of DLs which illustrate that DLs have limitations compared to the expressivity of FOL. The main reason for using DLs rather than first order predicate logic is that DLs are carefully tailored such that they combine interesting means of expressiveness with decidability of the important reasoning problems (Van Harmelen et al, 2008). DLs are decidable subsets of FOL where the decidability is due in large part to their having (a form of) the tree model property. This property says that a DL class C has a model (an interpretation I in which CI is non-empty) iff C (if and only if C) has a tree-shaped model, i.e., one in which the interpretation of properties defines a tree shaped directed graph (Grossof et al, 2003). OWL-DL is, therefore, a decidable fragment of first order logic, and thus cannot express arbitrary axioms: the only axioms it can express are of a certain tree structure (Motik et al, 2004). The is very relevant to railway domain development where the exercise is in providing a convenient means of representing domain information for its users rather than testing the boundaries of expressiveness and reasoning capabilities.

Decidability is a key concern in comparing the behaviour of reasoning functions. Decidability is the ability to entail a satisfactory conclusion from the available model and facts and is related to the expressivity of the logic representation. Decidability means that reasoning computations will finish in a finite amount of time (Lacy, 2005). While FOL is very powerful and expressive it is not decidable. DLs are decidable and, in theory, still powerful enough. One of the major challenges to implementing an OWL ontology design is managing the compromise between expressivity and decidable reasoning. The restrictions in OWL-DL are the same restrictions that make

reasoning systems decidable and are intended to support reasoning system requirements. Comparison of DL reasoners against FOL provers indicate that the latter systems can vary depending on the problem being solved. Therefore FOL provers may need to be used in the case where more complex DLs are required (Tsarkov and Horrocks, 2003).

Decidability is useful in the context of railway decision support as the intention is to test the model of the domain with instance data and perform reasoning over that data to assess the context in the data and any resulting actions required. In support of computational efficiency, OWL DL constrains the mixing of RDF and requires disjointness of classes, properties, individuals and data values. The advantage is that it enables reasonably efficient reasoning support and is therefore decidable. The disadvantage is that some of the compatibility with RDF is lost.

The purpose of the OWL DL sublanguage is to support tool builders that have developed powerful reasoning systems which support ontologies constrained by the restrictions required for OWL DL (Staab and Studer, 2010). OWL-DL extends the OWL specification by embedding Description Logic (DL) concepts that enable the capture and storage of logical constructs within the ontology model. OWL-DL promotes a descriptive mechanism of representing information within a domain. The challenge is that users of the system should not need to become experts in the modelling of ontologies in order to support system development. However, the resulting models need to be expressive enough to meet the various data requirements of the domain.

The design of DLs has been driven by the desire to provide practically useful knowledge modelling primitives while ensuring decidability of the core reasoning problems. To achieve the latter goal, the modelling constructs available in DLs are usually carefully crafted so that the resulting language exhibits a variant of the tree-model property (Vardi, 1996). This can prevent the accurate representation of structured, non-tree like, objects. The expressivity of DLs is often insufficient to accurately describe structured objects—objects whose parts are interconnected in arbitrary, rather than tree-like ways. DL knowledge bases describing structured objects are therefore usually underconstrained, which precludes the entailment of certain consequences and causes performance problems during reasoning (Motik et al, 2009). This feature is fundamental in the research - getting the right balance between the level of representation that is detailed enough to support the applications, and the level of constraint enabling satisfactory entailment leads to a higher likelihood of a successful ontology and knowledge store.

In the railway domain the structured objects are the components of the system, their internal configuration and their complex interrelationships with other components. These relationships extend to measurements and data related to those component parts. Motik et al present an equivalent example where a model of the human body, which consists of organs which can be decomposed into smaller parts. Complete decomposition will eventually lead to parts that one does not want or know how to describe any further. Representing this complete decomposition may in itself create an undecidable model which may require additional reasoning functions. In conclusion, the relationship between the level of model detail and the successful entailments presents a challenge to even simple model designs. The activity of ontology model

development will require a fundamental understanding of DL theory, which may preclude the contribution to model development by domain experts.

The temporal characteristics of an ontology may become an important feature in applications where temporal aspects play an important role. It is easy to imagine, in a large, active system such as the railway domain, how the timing of one event with respect to other events is important. In such cases the description logics of an ontology may involve temporal patterns. This subject has been the focus of a number of studies under the general title of temporal description logics (TDL) (Lutz et al, 2008). Some temporal characteristics may be imported from higher level ontologies or developed specifically to meet the requirement of the application. This therefore adds an additional burden on the designer to consider the temporal relationships of objects and relationships between them.

2.5 Semantic Web Technology

The *Semantic Web* supports semantic as well as syntactic data exchange by implementing ontology standards such as the Web Ontology Language (OWL) (Smith et al, 2004). The OWL specification is a specialisation of the Resource Description Framework (RDF), based on XML, and aimed at creating exact descriptions of things and their relationships (Kline, 2004). The OWL specification covers three flavours of progressively greater expressivity which are: OWL Light, OWL-DL, and Owl Full. That is to say that everything that can be represented in OWL lite can be represented in OWL DL, and everything that can be represented in OWL full can also be represented in OWL DL can also be represented in OWL full. The biggest distinction is between OWL full and OWL DL. In OWL full classes and properties are

individuals so we can represent classes of classes, classes of properties and classes can have properties. Furthermore, in OWL Full, literals are individuals while in OWL DL they are not. So in OWL full there is no real distinction between ObjectProperty and DatatypeProperty, while in OWL DL there is. The reason for these two flavours is that it is expected that some ontologies will be used in a way that allows them to be reasoned over by description logic inference engines (Colomb, 2007). Limiting the expressivity so as to make reasoning computationally tractable is a key aspect of ontology design and a matter that is addressed in the latter sections of this chapter. The specification of OWL-Full was devised by the OWL working group (Bechofer et al, 2009).

The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics.

OWL Full and OWL DL support the same set of OWL language constructs. The differences are in the restrictions on some of the features including some RDF features. OWL Full allows OWL concepts and RDF Schema to be mixed and, like RDF Schema, does not enforce a strict separation of classes, properties, individuals and data values. The advantage of OWL Full is that it is fully upward compatible with RDF, both syntactically and semantically. The disadvantage is that the language has become so powerful as to be undecidable which prevents any complete or efficient reasoning support (Staab and Studer, 2010).

In support of computational efficiency, OWL DL constrains the mixing of RDF and requires disjointness of classes, properties, individuals and data values. The advantage is that it enables reasonably efficient reasoning support and is therefore decidable. The disadvantage is that some of the compatibility with RDF is lost. The key purpose of the OWL DL sublanguage is to support tool builders that have developed powerful reasoning systems which support ontologies constrained by the restrictions required for OWL DL. OWL-DL extends the OWL specification by embedding Description Logic (DL) concepts that enable the capture and storage of logical constructs within the ontology model (Baader et al, 2003).

OWL Lite is a sublanguage of OWL DL which excludes enumerated classes, disjointness statements and arbitrary cardinality. The advantage is that it is easier to implement for tool builders, who want to support OWL, but want to utilise a simple basic set of language features. OWL Lite abides by the same semantic restrictions as OWL DL, allowing reasoning engines to guarantee certain desirable properties (Antoniou, 2004).

The ability to reason over data in order to infer new information presents the potential to realise opportunities for information management and decision support. For decades organisations have had the facility to perform some form of reasoning over existing data. This function was typically provided by a separate application, such as a rule engine forming an "expert system", and invoked at a point when all known information was available to the application (Liebowitz, 1995). The tacit knowledge of company experts is captured and stored for incorporation into business processes

without the explicit intervention of the domain expert. A separate application requiring a purpose built interface then enables the user to access the result of reasoning processes (Holsapple, 1998). The aim of the development of knowledge based systems was the integration with existing database technology in order to create 'enterprise level' knowledge management, which relies on the integration of knowledge management concepts into the existing corporate IT systems (Bolloju, 2002). In this sense, an advantage of a knowledge management strategy that implements a semantic approach, and is based on semantic technology, is that the reasoning function forms part of the information system and is always available to reason over the data currently available to it. However, the challenge to taking up this advantage is that it requires the deployment of a totally new way of storing and accessing data. As with any new technology, organisations are slow to adopt a new approach as it typically requires investment in order to deploy a new way of operating and therefore represents a risk to the business[2].

2.6 Middleware Concepts

The concept of middleware in Web based applications is very important but is the least visible in the application. In the context of the research, there are a number of alternative middleware options, but the actual chosen deployment is of little

[2] In the railway sector, *InteGRail* was a 20 million Euro EU funded project aiming to address the challenge of information system integration and interoperability (*www.integrail.info*). The formulation, storage and processing of tacit knowledge described in this thesis is fundamental to the approach proposed as a solution to the European railways.

importance. However, the middleware is important in supporting a complete understanding of the approach taken in this work.

There are a number of middleware programming languages such as Web Services, CORBA, Java RMI and DCOM. Web Services essentially form Web references that contain methods (functions) called by remote applications, regardless of their platform. Therefore, the power of Web services is that they support interoperability by machine-to-machine interaction over a network. Java RMI are more platform specific in that methods from remote Java objects (functions) can be called from other Java virtual machines. However, the principle for most of these middleware is the same. The important point is that the middleware is a mechanism for transferring a message from one place to another. What is performed as a result of that transfer is independent of the middleware and highlights the requirement for a generic machine interpretable model, i.e. there is greater benefit to be gained through having dynamic, generic data processing applications operating across an open network of nodes as compared to having static, data model specific applications.

2.6.1 Reasoning Applications

The purpose of reasoning in the context of this thesis is to enable the link to be made between the ontology and its intended meaning. A reasoner application also provides important consistency checking of ontology models and the defined content (Baclawski et al, 2002). This is less interesting in the context of this thesis though important in the design process. The foundation of the RDF syntax on XML standards and the concrete data types allowed in OWL enable instances of data to be stored. Reasoners enable some interpretation to be made and therefore some context to be

assessed (Dey, 2001). This is relevant to the proposed requirements of the railway domain because there is a need to recognise the context on the data, e.g. what asset is was measured on, the location and what it was measured using. This provides useful information when data sets need to be combined to make a decision on a particular course of action. This concept can be extended to modelling situation awareness for decision support which is perhaps the more advanced application of embedded description logics (Feng et al, 2009). The usefulness of the context depends on the models that were created in the first instance prior to the population of the repository and the application of the reasoner. This in itself is a subject matter covering a wide area of research (Motik et al, 2006).

The purpose of implementing a DL reasoner is to generate the interpretation of the available facts against the ontology concepts modelled. This is when the decidability discussed introduced in Section 2.5 becomes important as the facts introduced must produce a result that is satisfiable in the domain. In the context of railway systems interoperability the reasoner is proposed as fundamental feature of the applications that share the ontologies and exchange information. The challenge is two fold:

- Models must be created that represent the information that the stakeholders want to exchange
- Architectures and applications must be developed to exchange the data and present the information relevant to the context

In summary, the relevance of the reasoning function to the railway domain ontology design is that it enables the designer to check the consistency of the model and statements during the design process. For the proposed applications the reasoner is relevant because it provides the ability to extract model excerpts by querying, reason

over the result that has some potential inferred meaning and create some new, explicit information. The aim of the thesis was to attempt to provide examples of where this implicit – explicit translation can create value.

There are two examples introduced in this thesis: 1) when two different systems predict an event (the way the system presents the information may be different but the resulting effect on the decision support system is the same). This is in effect catering for the heterogeneity in data between systems. 2) The second case is where there are complex information processes underpinning complex management tasks. These scenarios are common in large scale systems such as railway operations and the latter sections of the thesis aim to address these requirements.

2.6.2 Non-monotonic reasoning

A monotonic formal logic is one in which the addition of a formula or query does not reduce the set of consequences. Ordinary deductive reasoning is monotonic because anything that could be concluded before a clause is added can still be concluded after it is added. A logic is non-monotonic if some conclusions can be invalidated by adding more knowledge. A non-monotonic system is one in which learning of a new fact or piece of knowledge enables the inference of a new consequence. Etherington, Kraus and Perlis propose the implementation of non-monotonic reasoning in reaching conclusions which are not strictly entailed[3] by what is known. It is proposed that the conclusion of a set of facts can be altered when new information is added (Etherington, Kraus, & Perlis, 1990).

[3] Entailment: a logical relation between sentences of a formal language.

OWL-DL reasoners implement non-monotonic reasoning and therefore enable a mechanism where instance data results can be added, assessed for a particular result and later appended with additional information and reassessed. This functionality is important in the development of a semantic application where a number of resources are queried and the results are incrementally appended and reasoned over as they become available.

2.6.3 Storage of Ontology Models and Data

The mechanism for storing ontology models and instance data is independent of the ontology representation and design. One aim of the research was to demonstrate to industry stakeholders that existing tools could readily be implemented to build a working prototype. Therefore the Jena Application Programming Interface (API) was implemented as a mechanism for storing and manipulating the ontology concepts and instance data (Carroll et al, 2003). However, there are other mechanisms for model storage and manipulation and the ISO15926 is an example that was analysed during the research (ISO15926). Jena stores data in a triple store where data is represented as a *Subject – Object* pair that is connected by a *Predicate* attribute:

<SUBJECT> → <PREDICATE> → <OBJECT>

A triple store is a file-based system or relational database system that has a database schema and retrieval mechanisms optimized for triples. Triple stores also make provision for reasoning capabilities, such as inference and querying. Triple stores can be contacted either by using an API, directly inside the application, or using a remote

communication method, such as SOAP (Simple Object Access Protocol), HTTP (Hyper Text Transfer Protocol) or RMI (Remote Method Invocation).

Two basic architectures are used in the development of triple stores, as shown in Figure 2.1. The difference between them is found in the scope of the reasoning and inference supported and in the modularity of the triple store. In an application that does not require any heavy inference to be undertaken, a light-weight triple store which does not have any added inference engines for supporting the retrieval of information can be implemented. However, where there is a requirement to infer information to support information retrieval, fully-fledged repositories can be implemented. These are modular in nature and provide facilities for tuning the triple store towards the application's needs. External reasoners or inference engines can also be linked to the repository. Another important aspect is the ability to link up a variety of external databases, instead of using a proprietary internal format.

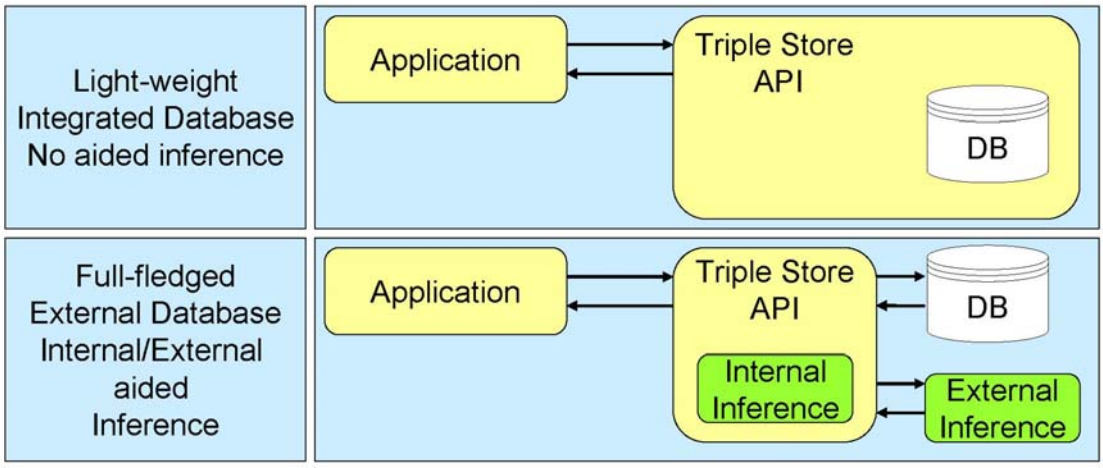


Figure 2.1 Two types of triple stores

In the triple-store arrangement a distinction is made between the descriptive aspect of the repository, the 'terminology model' (T-Box) and the instances of information

associated with those descriptions, called the 'asserted model' (A-Box). The T-Box describes the system by means of classes and relationships. The A-Box, on the other hand, represents the instances of the corresponding T-Box concepts. A-Boxes are T-Box-compliant statements about the vocabulary that they use and together T-Box and A-Box statements make up a knowledge base. The proposition that *"an ontology is an explicit specification of an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly"* fits well with this T-Box and A-Box arrangement (Gruber, 1995). A classical comparison used in this context is one in which the T-Box concepts are mapped onto the classes in Object-Oriented programming, while the A-Box instances correspond to the run-time objects of the classes in Object-Oriented programming. The statement in Table 2.3 illustrates this description.

Table 2.3 A-Box and T-Box representation

A-Box	T-Box
A is an instance of B Vehicle with ID 1234 is of type Diesel Engine Wheel_3 hasObservation Obs_1	An Engine has two subclasses; Diesel and Electric A Railway Vehicle has 8 Wheels

The RDF and OWL languages provide the standards for creation and definition of semantic models, but introduction and combination of these models with existing legacy systems is not clear. This is an important issue for the railway industry as there are numerous legacy systems still in use. One potential solution is the D2RQ Mapping Language. In D2RQ non-RDF databases are treated as virtual RDF graphs

(Bizer and Seaborne, 2004). This method utilises a language capable of expressing mappings and links between tables, rows and columns of the relational database, to the concepts, individuals and relationships of the semantic model. This results in a 'virtual' triple store, based on an existing relational database and used and populated by existing legacy systems. The proposed architecture for using this mapping language is one with a single central relational database. The benefit of this architecture is that a legacy application can populate and retrieve data to and from the relational database as in the conventional manner while concurrently plugging in a semantic application and using the mapping language to create a virtual triple store. This approach brings with it the many advantages that come with the ability to reason over the semantic data created at run-time from the non-semantic data in the relational database. This means that support is injected into the overall systems at run-time even though there is no native support for semantic modelling and reasoning. This virtual semantic data in turn results in the ability to infer new knowledge about the system.

2.7 Ontology Representation

In this thesis the Web Ontology Language (OWL) is used as a basis for exchange of messages between interoperating systems in conjunction with Description Logic (DL) reasoners such as Pellet, which are implemented to infer some meaning from the content (Sirin et al, 2007). An important feature of the implementation of the concepts is the model theory which divides a data model into two parts: the formal meaning of the theory, and the intended interpretation of the theory (West, 2011). In description logics the intended interpretation is any non-empty set of objects and an interpretation mapping from the non-logical symbols to functions and relation over the set

(Brachman and Levesque, 2004). If a particular set of data can be inferred to satisfy a specific model theory then the implicit content is made explicit.

Early work on Web based knowledge representation sought to build on the existing model theory of the RDF Schema specification (Hayes, 2004), (Broekstra, 2001). The RDFS model theory is represented as domains and mappings of interpretation I, basic equations, class extensions, property extensions and domain and range restrictions. OWL Full has been given a model-theoretic semantics that is a vocabulary extension of the RDF model theory (Patel-Schneider et al., 2004). A correspondence between this semantics and the semantics of OWL DL has also been established: it has been shown that the model theory for OWL DL has very similar consequences to this RDF-style model theory for those OWL ontologies that can be written in the OWL DL abstract syntax (Horrocks et al, 2007). The OWL-DL model theory is represented by vocabularies and interpretations arising from the embedded syntactic constructs, axioms and facts provided by the extensional semantics (Patel-Schneider and Horrocks, 2004).

Prior to defining any ontology design concepts, it is necessary to decide on a particular annotation to apply during the design. Since UML is specific to classic object oriented classes which contain concepts such as methods as well as other attributes that do not fit with OWL classes, a simplified approach was taken. Figure 2.2 illustrates the representation of the triple with the associated HTTP URI reference. Information is described by nodes and edges represented by Uniform Resource Identifiers (URI), where nodes are represented as subjects and objects and edges are

represented as predicates. A "triple" represents a unit of information by diagrammatically combining a subject, a predicate and an object.

Figure 2.2 Subject predicate object RDF triple

The Resource Description Framework Schema (RDFS) is an ontology language written in RDF that supports the specification of resources and properties, sub classes, super classes with inheritance, the instantiation of types and domain and range property restrictions (RDF Vocabulary Description Language, 2004). Figure 2.3 illustrates how these restrictions are created by the associated "rdfs:Class"[4], where the concepts "Component" and "Fault" are "subclasses" of "rdfs:Class". This means that they inherit all of the features of this class. The "rw:hasFault" property is a "subPropertyOf" "rdf:Property"[5]. This means that it inherits all of the features of the "rdf:Property". The "rw:System" illustrates how concept hierarchies are defined.

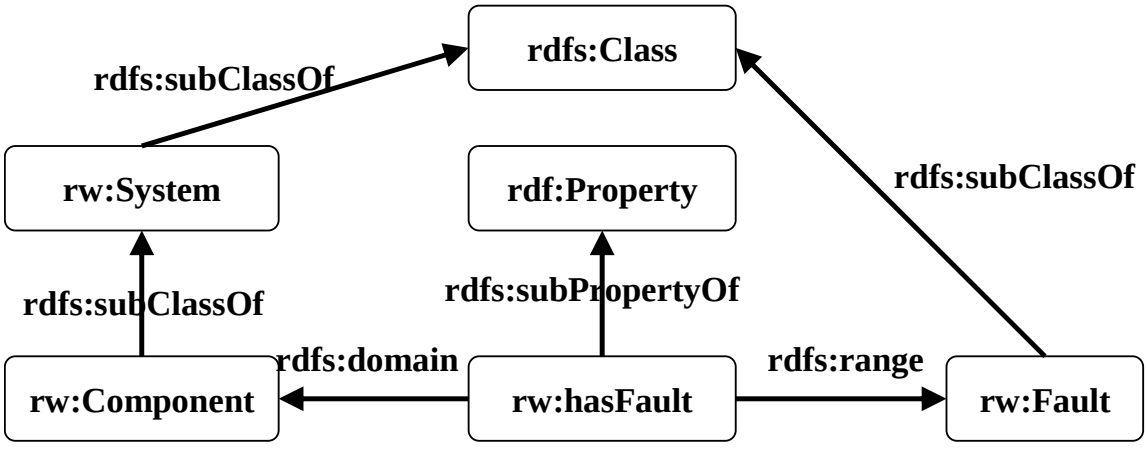


Figure 2.3 Association between RDF classes and RDFS restrictions

[4] RDFS refers to the abbreviation of URL – Resource Description Framework Schema

[5] For convenience, the prefix URL *http://www.integrail.info* has been abbreviated to *rw.*

Figure 2.4 shows the shorthand notation of OWL ontologies, enabling a more concise representation of the triple shown in Figure 2.2.

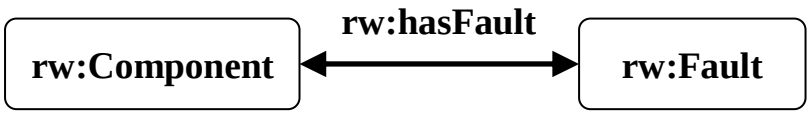


Figure 2.4 Shorthand triple representation

The principle of ontology design is on re-use and extensibility of existing ontology resources. Supporting the design of a railway domain ontology requires the review of a vast array of existing ontology resources to establish if any are of value and therefore to be re-used. To put this in context it is necessary to consider the type of information required to be transferred across a network of railway applications.

For the effective retrieval of train information, it is frequently necessary to associate train observations with some spatial information. SOUPA, the standard ontology for pervasive and ubiquitous computing is a shared model that represents a number of concepts for use in semantic applications (Chen et al, 2004). Figure 2.5 illustrates the specification for a spatial model called Region Connection Calculus (RCC) used in specifying the spatial relationship between model concepts (Cohn et al, 1997). There are many other existing ontology resources that are readily available for import into a project.

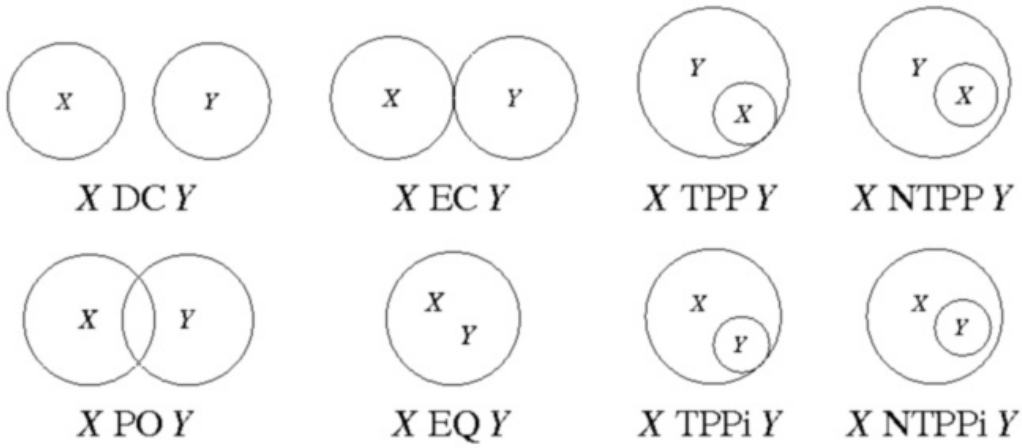


Figure 2.5 Region connection calculus spatial models

These spatial models refer to disconnected (DC), externally connected (EC), tangential proper part (TPP), non tangential proper part (NTPP), part of (PO), equal (EQ), tangential proper part inverse (TPPi) and non tangential proper part inverse (NTTPi) concepts. To meet the requirements of the railway domain it is necessary to extend these existing models to make the concepts railway specific rather than general. Figure 2.6 represents the model for 'Railway Territory' concepts; note that the prefixes of 'RCC:' and 'Railway:' are used to indicate that the model is in fact a specialisation of RCC for the railway domain.

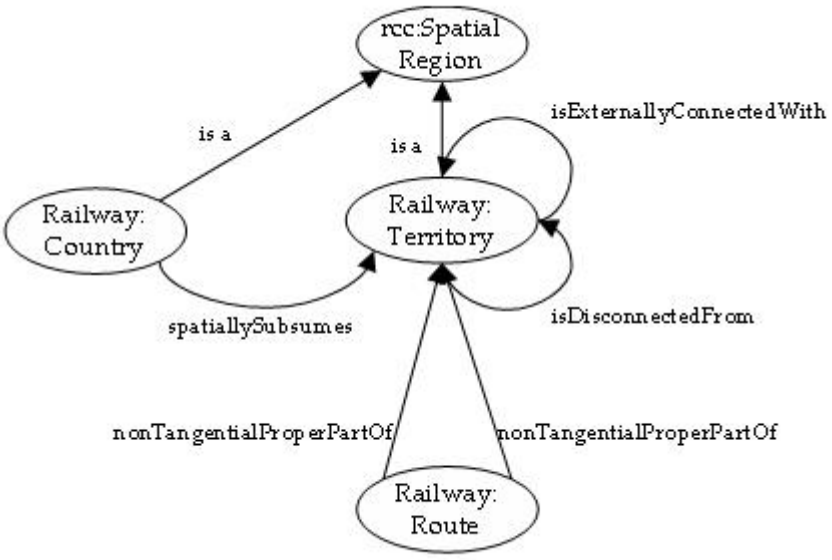


Figure 2.6 Spatial Relationships for the Railway Domain

Clearly there are many different *types* of assets and components in the railway domain. For example, there are many different types of train fleet. However, train maintenance companies are usually only interested in gathering data for one type of train fleet, centralizing that data at a location such as a maintenance depot. Information systems handling fault data are therefore only concerned with one type of fleet data but the requirements for broader integration, such as those of an integrated European railway network, will focus on data from a greater array of train types. This presents a challenge as not all train subsystems are identical, and new trains may represent a new requirement for information storage. One way of dealing with this heterogeneity in data is to define design patterns, as proposed by Gangemi (Gangemi, 2005). In this way, the ontology is considered to be part of a lifecycle of system development and therefore grows with the demands of the real world system. This is an important aspect of the ontology design process described in further work in this thesis.

2.8 ISO 15926 Standard

This standard is titled "*Industrial automation systems and integration—Integration of life-cycle data for process plants including oil and gas production facilities*". It is a standard for data integration, sharing and exchange between computer systems. The standard was developed in the 1990s and aimed at representing data concepts associated with plant within the Oil and Gas industry. This standard has since been proposed for use in the wider community (ISO15926).

The standard consists of 7 parts:

- Part 1 - Introduction, information concerning engineering, construction and operation of production facilities is created, used and modified by many different organizations throughout a facility's lifetime. The purpose of ISO 15926 is to facilitate integration of data to support the lifecycle activities and processes of production facilities.

- Part 2 - Data Model. A generic 4 dimensional model that can support all disciplines, supply chain company types and life cycle stages, regarding information about functional requirements, physical solutions, types of objects and individual objects as well as activities.

- Part 3 - Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 3: Reference data for geometry and topology.

- Parts 4,5,6 - Reference Data, the terms used within facilities for the process industry.

- Part 7 - Implementation methods for the integration of distributed systems, defining an implementation architecture that is based on the W3C Recommendations for the Semantic Web.

The data model represented in Figure 2.7, referred to as the *lifecycle integration schema*, supports the creation of meta-level concepts. In standard ISO15926 these concepts are a 4D upper ontology – an extensible set of statement types that can be made in that context. These concepts are recorded in class diagrams defined using ISO 10303-11 EXPRESS which is itself a generic data modelling language. The current standard has been used to specify *reference data libraries* for the oil and gas industry and therefore requires work to be undertaken to specialise model concepts for other domains. Classes of this standard are not equivalent to classes in the UML representation or the Web Ontology Language standard, though there are some similarities. Members of classes can be individuals, relations or other classes. In OWL, only concepts are referred to as classes and therefore only individuals can be members of classes. Figure 2.7 illustrates the highest level concept the ISO 15926 lifecycle integration schema or data model.

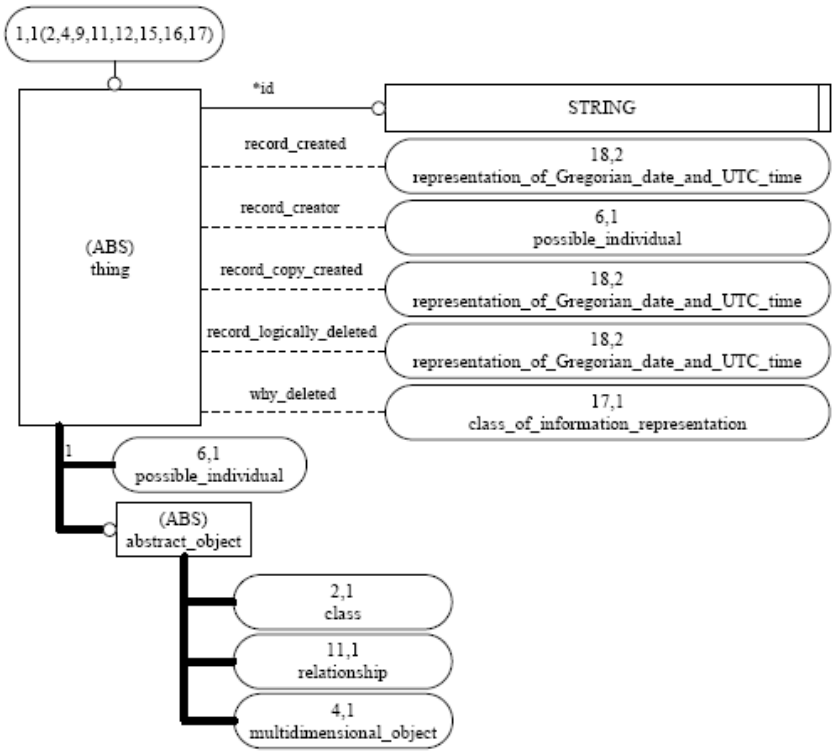


Figure 2.7 Highest level concept including subtypes and attributes taken from ISO 15926

The advantage of the ISO 15926 approach is that both the class model and data are optionally stored in a number of formats including a relational database format. This is in contrast to standards based on XML which require transformation into a standard database representation, or mapping to object oriented objects. The inverse of this is that any requirement to interchange data openly over the internet will lead to a transformation of data into an appropriate structure and format. In the interest of open data interchange many sectors of business are favouring the use of common standards for information interchange. Figure 2.8 represents an example reference data model for a parts list.

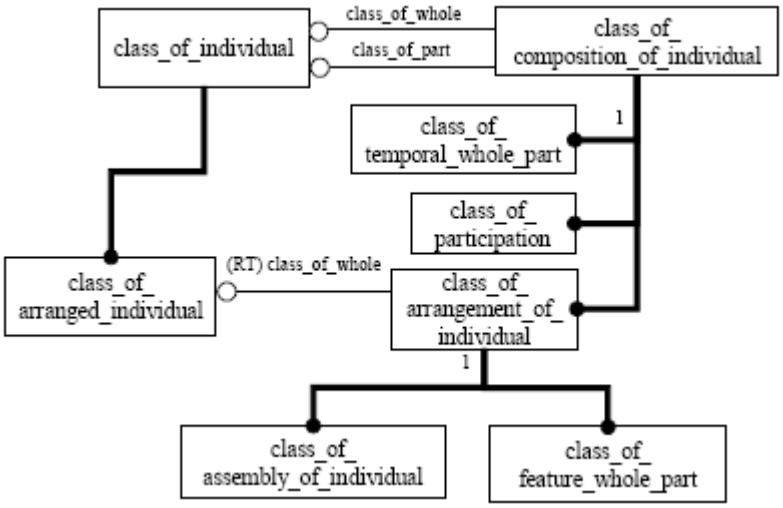


Figure 2.8 Ontology level reference data model for a parts list

The ISO 15926 data model is designed to be used in conjunction with reference data: standard instances that represent information common to a number of users, plant or processes. The Part 2 data model is specified using the EXPRESS language described in ISO 10303 and can be represented graphically using the EXPRESS-G notation. The EXPRESS language is in an ASCII format that shares similarities with the Pascal programming language. ISO 10303 is a standard for the computer-interpretable representation and exchange of industrial product data – known as STEP or Standard for the Exchange of Product model data (ISO 10303). The format of the step file is defined in ISO 10303-21.

In order for instance data to be referenced to the meta-level data model, an application is required. This application is available in the form of an API (Application Programming Interface) which can be imported into the development platform (C, C++, Java). ISO 10303-22 is a part of the implementation methods of STEP with the official title Standard Data Access Interface or SDAI. SDAI defines an abstract Application Programming Interface. JSDAI is an example of an API for reading

writing and runtime manipulation of object oriented data defined in an EXPRESS data model.

ISO 10303-28 is an XML representation of the EXPRESS schema and data. This does not share the capability of capturing the semantics of the express schema with the OWL language.

2.9 ISO 13374 - MIMOSA

This standard is named *Condition Monitoring and Diagnostics of Machines – Data Processing, Communication and Presentation* (ISO13374). The aim of the standard is to overcome the interoperability issues associated with condition monitoring IT systems that restricts the unified view of machinery to its users. This standard is implemented by MIMOSA, the Machinery Information Management Open Standards Alliance, in the Open Systems Architecture for Condition Based Maintenance (OSA-CBM – see www.mimosa.org).

The standard is different to the other approaches because it is focussed on creating a standard architecture for deployment of a CBM system. This deployment is based on a predefined UML data model that can be optionally implemented as an XML Schema. This means that the storage required can be easily configured to match elements of the XML schema. The standard describes six functional blocks of CBM architecture that can be optionally deployed, as shown in Figure 2.9. In principle, this standard defines the interfaces between the functional blocks and information

exchanged between them, leaving the developer the important task of creating code that responds to the messages received.

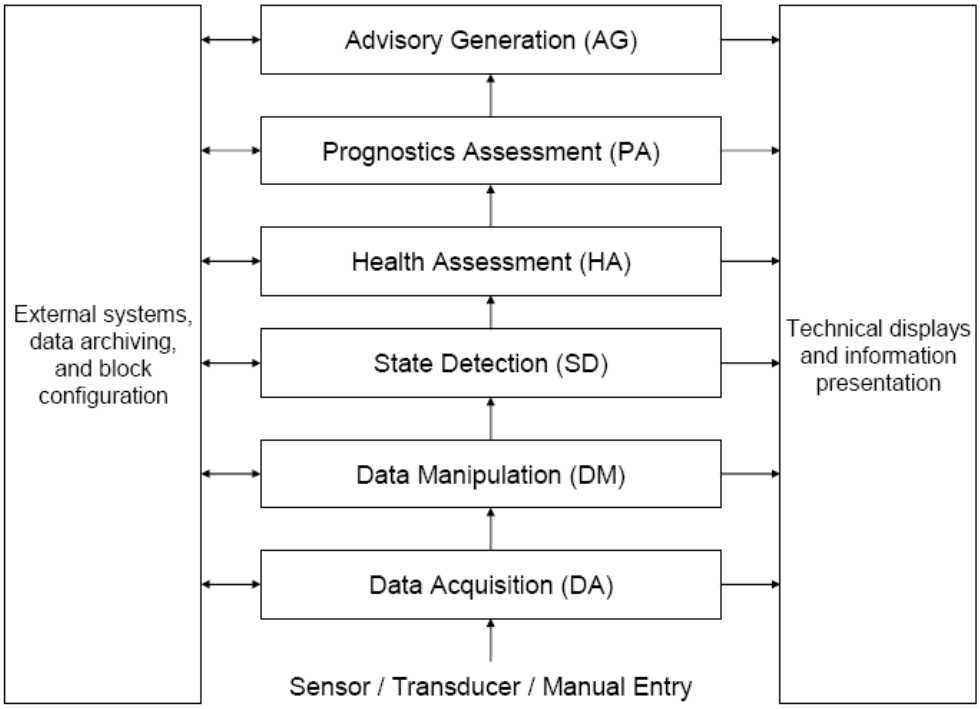


Figure 2.9 ISO 13374 data processing and information flows

The semantics of the data objects captured in the UML model are defined in written English descriptions and recorded in a terminology dictionary, but this is not part of the standard. There is of course scope to extend the data model with semantic models based on the RDF and OWL standards. Again, this is not defined in ISO 13374 .

2.9.1 Example

One approach for the deployment of this standard is based on Web services. Each functional block is deployed as a Web service that is contactable by all other blocks and optionally contacted by external Web services, as shown in Figure 2.10. Using this approach, data is exchanged over the Web service by XML, which is then decoded/manipulated and passed to the individual application in the block. Each application or algorithm can be written in proprietary code that is invisible to the rest of the network.

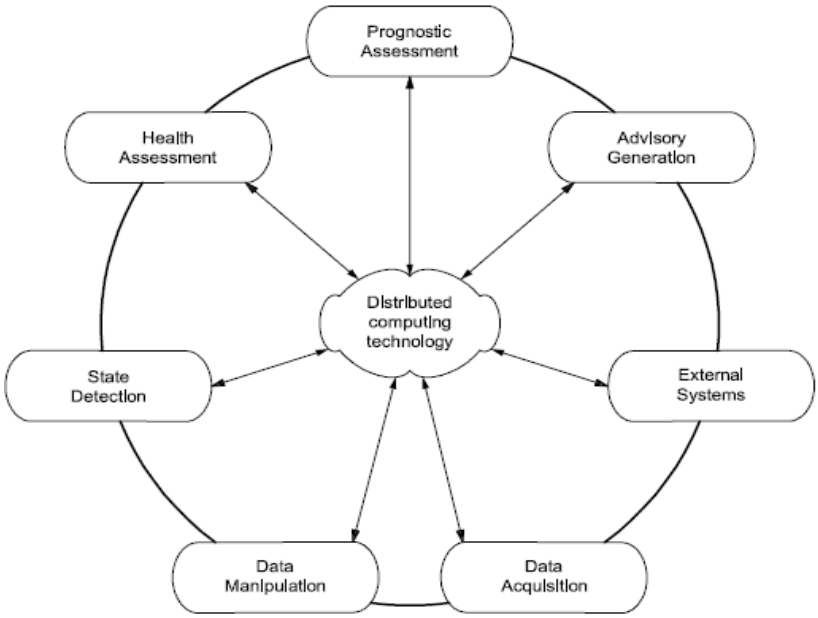


Figure 2.10 Data processing in the standard architecture

2.10 Discussion

Architecture

- The OWL standard does not have any particular architectural components associated with it. OWL is typically associated with open source platforms deployed over a SOA (Service Oriented Architecture) and therefore is aimed at Web applications that have a greater focus on the semantics of data exchanged.

- ISO 15926 also does not have specific architectural requirements, though there is an XML derivative of the existing oil and gas reference data library. It is proposed that this XML schema can be implemented in the exchange of XML data over a Web service.

- ISO 13374 has a well defined architecture, with defined interfaces and a strong data model – but the semantics of the data model are not naturally machine interpretable.

Data

- The OWL data model has the namespace extensibility of XML with the benefits of enabling truly interoperable data to be produced i.e. if two applications share an ontology and possess the same reasoning application, they can infer the same result. This is achieved without additional

manipulation of the exchanged data. The down side is that all data needs to be stored in a triple store, and hence in triple notation, to achieve this.

- ISO 15926 data can be exchanged by XML or OWL over a web service. However, this requires the formatting of the data into the appropriate syntax for exchange. The receiving system is then required to decode the data back into the structure that the ISO 15926 data model is built on. The strength of this approach is that the data is still referenced to a hierarchical ontology model, which should provide a rich semantic of the received data once it has been decoded.

- ISO 13374 is much weaker from the data modelling perspective, offering a predefined schema and limited semantic capture. The plus side is that the standard approach is readily deployable across a distributed architecture.

Deployment

Figure 2.11 represents the hierarchical view of the OWL and ISO 15926 ontology structures. This diagram highlights that there are parallels to the two approaches to modelling data.

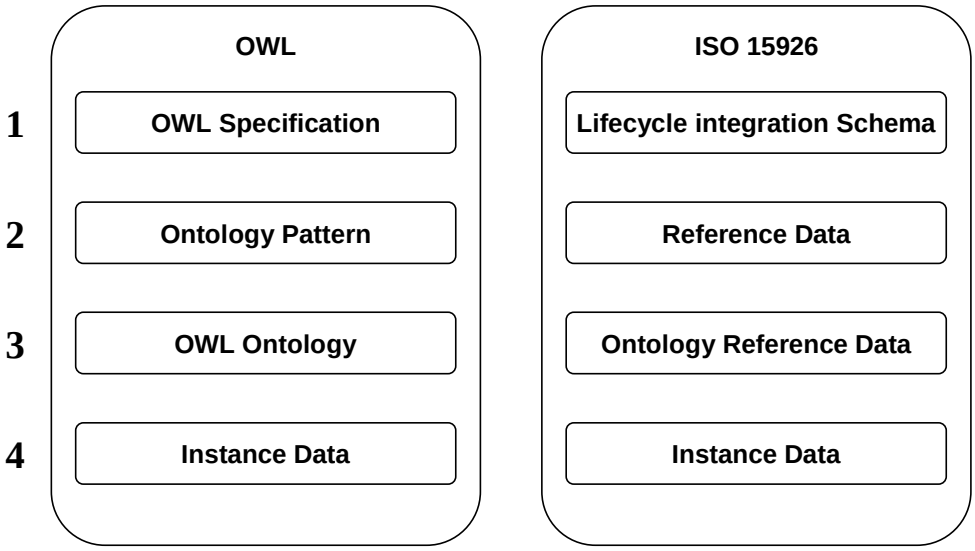


Figure 2.11 OWL and ISO 15926 ontology modelling hierarchies

Level 1 represents the highest level of the model. This is a layer that dictates how things such as classes, attributes and properties can be arranged in the lower levels. This essentially represents the rules for ontology construction

Level 2 represents some domain specific specialisation using the rules defined in the layer above. They effectively represent the generalisation of the ontology models defined in level 3. In OWL this is achieved through the creation of a number of reusable ontology design patterns. In ISO 15926 this is achieved through the creation of one or more *reference data library.*

Level 3 represents a domain and implementation specific specialisation of the model defined in the layer above. These models literally contain concepts that represent real world devices such as pumps and motors and the attributes of the devices such as temperature and speed. The modelling concepts are implemented by the layer below to record real world instances of those devices and data recorded about those devices.

Level 4 represents that layer that typically stores data referenced to the layer above. In the case of an OWL ontology, the reference is made by an OWL reasoner such as Pellet. For ISO 15926 ontology this reference is made through the implementation of runtime applications such as JSDAI discussed earlier.

ISO 13374 does not consider the modelling requirements as the UML/XML model is predefined. However, the standard architecture of ISO 13374 could be extended by either of these modelling approaches to capture the semantics of the data.

While there are some significant differences in these approaches which are outside the scope of this discussion, there is some overlap. The implementation of any of these approaches would be dictated by the requirements of the application. Table 2.4. outlines a comparison of some potential deployments.

Table 2.4 Comparison of interchange standards

	OWL	**ISO 15926**	**ISO 13374**
B-2-B applications (such as product inventory/stock control)	Already implemented for Web based transactions such as sales	Data design is restricted to a relational data model – this then needs to be formatted as XML or OWL	Architecture too rigid for direct implementation
Enterprise Applications (such as Network Rail intelligent infrastructure)	Feasible, but large overhead to store data as triple data – conversion is required for legacy data	Appealing, as data is stored and referenced in database format – conversion would still be required to map legacy data to defined ontology model data	Simple solution but no semantics are captured – could result in a large repository with little inherent context between data elements
Highly	Appealing, as	The approach	Feasible, but does

geographical heterogeneous systems (such as European railways)	heavy or light ontology implementations can be created to suit various open message interchange. Ontology can be specialised by each partner and only necessary data converted from legacy system to OWL (shared) format.	proposes that all data is formatted to match database ontology so there is still some overhead for legacy systems. Requires data models to be mapped to XML or OWL for open internet exchange.	not offer any more than a convenient way of exchanging messages. It does not tackle the issue of knowledge management based on data from multiple complex systems.

2.11 Conclusion

A review of concepts associated with decision support considered in this chapter is set in the context of semantic technologies. This includes the definition of ontology concepts and the description of applications associated with the deployment of ontologies in knowledge management applications.

Conventional methods of collecting and storing data using relational databases are limited to direct insertion and retrieval of facts. Any inherent meaning is limited to the relations between data terms. In contrast to this, a methodology that integrates the meaning of terms within the data model is available to organisations that rely on knowledge as a commodity. The computer science community have adopted the term *ontology* to represent this knowledge model along with Semantic Web tools, the generic term for applications capable of interpreting and processing those ontologies.

Technologies and tools developed for the Semantic Web have potential for solutions in industrial data interchange and integration applications such as decision support. By adopting modern tools for storage, transport and interpretation of ontology based knowledge concepts, the railway domain could move to a new era of domain knowledge management that supports many facets of operation, including maintenance management and diagnostic and prognostic processes.

In the next chapters attributes of the OWL, ISO 15926 and ISO 13374 approaches to large scale system integration are drawn upon to develop a solution suitable for the railway domain.

CHAPTER 3

KNOWLEDGE MODELLING AND ONTOLOGY DESIGN

3.1 Introduction

There are two major challenges to knowledge capture and modelling for decision support. The first is the action of collecting and collating knowledge from the domain expert; this is challenging because people are not familiar with recording their ideas in a particular format. Therefore, attempting to gather knowledge of a domain through engagement with domain experts requires some mutual understanding of the process. Stakeholders may be used to recording information but this will not necessarily constitute knowledge. The second major challenge is the action of storing and using knowledge in a manner that is convenient for deployment on an industrial scale.

This chapter introduces the idea of an ontology design process that is simple enough for rapid uptake yet sufficiently complex to support the retrieval of useful information and also the capture of context within it.

3.2 Ontology Design

Meeting the needs of stakeholders within a large scale integration project requires management of the ontology design process. The priority is to ensure that the stakeholders are involved in the model definition process and that this process meets their needs. In addition, the ontology design process must at least meet the requirements for an appropriate interchange format for data integration, containing mechanisms for recognising context in railway information derived from that data. Dealing with these competing demands is difficult in a multi-disciplined environment; the use of some design constraints is vital and an ontology methodology is a necessary supporting tool (Cristani and Cual, 2005).

In knowledge modelling, the parallel activities of addressing the scope of the kind of data interchanged and integrated and the kind of querying performed forms the foundation of the ontology design. On achieving a fundamental understanding of the domain, a follow on activity defining the *types* of model patterns is undertaken and these design patterns support the specification of 'core' ontology concepts. Doerr proposes that "a core ontology is one of the key building blocks necessary to enable the scalable assimilation of information from diverse sources" (Doerr et al, 2003). A complete and extensible ontology that expresses the basic concepts that are common across a variety of domains and can provide the basis for specialization into domain-specific concepts and vocabularies is essential for well-defined mappings between domain-specific knowledge representations and the subsequent building of a variety of services such as cross-domain searching, browsing, data mining and knowledge extraction". Staab et al. describe the creation of semantic patterns and their comparison with software design patterns; an approach which lends itself to domain

knowledge capture by enabling the domain expert to incrementally specialise the component patterns in the model (Staab, 2001). The patterns result from four design criteria:

- representing the sequential relationship between physical components;
- representing the abstract dependencies between them;
- representing their relationship with external observations;
- representing mappings to concrete data types, such as legacy data, for historical analysis.

Figure 3.1 illustrates the general pattern for sequential component modelling. This pattern shows the representation of physical railway concepts such as 'tracks' and 'routes' and that the connection 'points' links them together. These physical concepts are generalised to higher level concepts called 'ComponentEdge' and 'ComponentNode'. At this level of the design process the aim is to create a set of concepts which are as general as possible so as to be useful to the railway domain. This collection is used to specialise all the concepts required to capture information about the domain; the collection is referred to as the *core* ontology.

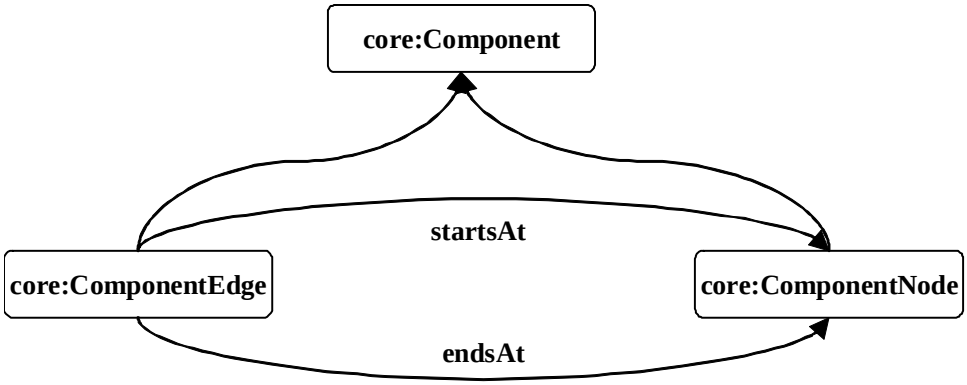


Figure 3.1 Sequential component design pattern

To represent concepts of a railway train, a second pattern represents the relationship between vehicles. Figure 3.2 illustrates a pattern for linking a train to one vehicle through the *hasFirstVehicle* property, linking subsequent vehicles through the *hasNextVehicle* property.

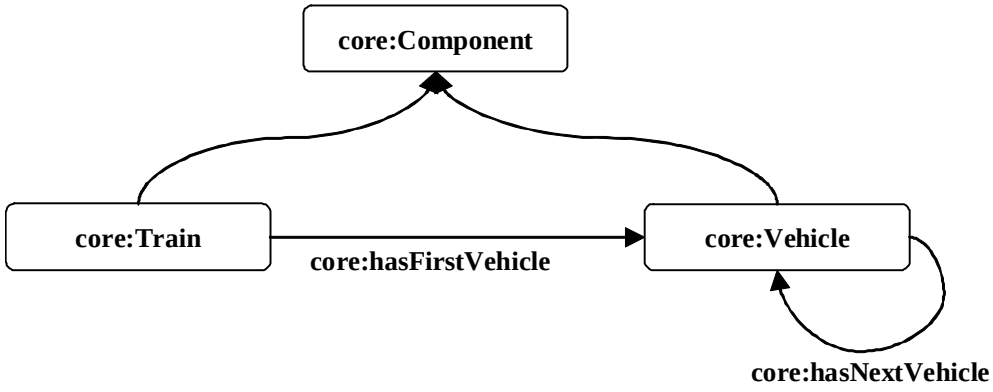


Figure 3.2 Railway train design pattern

A general requirement for the core ontology is the ability to attach measurement concepts and some overall status condition to physical component abstractions. This feature brings the model into alignment with existing legacy system design, which

typically represents data in a standard relational database, associating some measurement with some asset record (Spyns, 2002).

The concept of Model Theory was introduced in Chapter 2 section 2.7. The separation between the model theory and its interpretation is equivalent for both conventional data modelling and the ontological equivalent. The main difference is that ontology concepts in the context of the Semantic Web are aimed at creating machine interpretable content. The interest from the railway industry perspective is in creating applications that capture context and interpret the implied content into explicit information. This concept is not impossible to achieve in conventional data base modelling. However, in the railway domain the focus was on capturing the tacit knowledge of industry experts in a manner that could be incorporated into ontological concepts. Therefore one aim was to demonstrate that situations where information was incomplete or inconsistent could still lead to useful outcomes i.e. the information that was implicitly represented could be reasoned over to create some explicit conclusion. Since the industry experts are not experts in database or knowledge modelling, there was a need to create a process by which information could be acquired and modelled. However, the ontology design differs from this conventional way of system development. A proposed benefit of an ontology based design is that there is an opportunity to capture some implied content which is later made explicit through consideration of all concepts associated with a particular scenario. A system that only handles explicit content is said to be closed world and as such utilises the C*losed World Assumption* (CWA). CWA posits that what is not known to be true is considered to be false. A database is an example of an information system that implements the closed world assumption because any information that is not

explicitly stated is interpreted to be negative. A system that can handle implicit content is referred to as open world and as such utilises the O*pen World Assumption* (OWA). OWA posits that what is not known to be true is only considered to be false if it contradicts other information in the knowledge base.

Damiani demonstrates the concept of OWA through a simplified case study referred to as the *Little House Problem*. This is followed by an example of a real world information system problem where there is need to integrate heterogeneous representation schemes and reconcile the concepts within two systems (Damiani, 2005).

An understanding of the OWA concept is important to put the work described in this thesis into context. The example provided by Brachman and Levesque demonstrates the syntax of the First Order Logic* (FOL) applied to a simple reasoning problem – the "blocks-world" example (Brachman and Levesque, 2004). Figure 3.3 represents three coloured blocks stacked on a table. Suppose that it is known that the top block is green and that the bottom block not green, and the colour of the middle block is not known. The problem to be solved is to identify if there is a green block directly on top of a non-green one. The answer, which happens to be yes, is not immediately obvious without some thought by the reader.

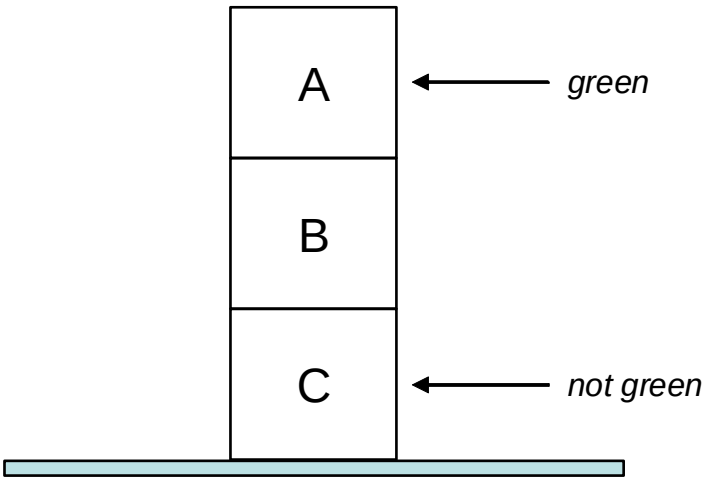


Figure 3.3 Model Example – a Stack of Three Blocks

The problem is formulated in FOL where *a, b* & *c* are the names of the blocks and predicate symbols *G* and *O* represent "green" and "on". If the set of sentences representing the information about this domain is *S*, then facts currently held in *S* are:

$$\{O(a, b), O(b,c), G(a), \neg G(c)\} \quad \text{- (1)}$$

It is postulated that these four facts provide enough information to *entail*[†] that there is a green block on top of a non green one, i.e. that α is satisfied in the knowledge base *S* and where α is represented by:

$$\exists x \exists y.(G(x) \wedge \neg G(y) \wedge O(x, y) \quad \text{- (2)}$$

Which reads as: there is "at least one" variable *x* and "at least one" variable *y* where *x* is *green* and *y* is *not green*, and *x* is directly on top of *y*. To prove that this is true it needs to be shown that any interpretation that satisfies *S* also satisfies α. Letting $\mathfrak{I}$ be any interpretation and assume that $\mathfrak{I}$ satisfies *S*. There are two cases to consider here,

the first is the case where the middle block is green – $\mathfrak{I} \models G(b)$. In this case, because $\neg G(c)$ and $O(b, c)$ are in S,

$$\mathfrak{I} \models G(b)\ \neg G(c) \wedge O(b, c). \quad \text{- (3)}$$

And it follows that

$$\mathfrak{I} \models \exists x \exists y.(G(x) \wedge \neg G(y) \wedge O(x, y). \quad \text{- (4)}$$

i.e. $\mathfrak{I}$ does indeed satisfy α in this case.

Consider a second case where the middle block is not green, $\mathfrak{I} \models \neg G(b)$, because $G(a)$ and $O(a, b)$ are in S,

$$\mathfrak{I} \models G(a)\ \neg G(b) \wedge O(\text{a, b}). \quad \text{- (5)}$$

And it follows that

$$\mathfrak{I} \models \exists x \exists y.(G(x) \wedge \neg G(y) \wedge O(x, y). \quad \text{- (6)}$$

i.e. $\mathfrak{I}$ does indeed satisfy α in this case.

This simple example serves to demonstrate that, without explicitly stating the colour of block b, some reasoning process is relied on to establish that it is logical for one of the outcomes to be true, thus making the statement represented by α true in S. This means that implicit content captured in this information can generate further explicit information if reasoned[‡] in the appropriate manner.

The "blocks- example" serves to demonstrate that a context aware system could be developed that caters for the variability in the way data, and therefore information, is represented. In support of the argument to follow a semantic approach to knowledge management, a similar approach is taken in the context of information system integration where there is need to cater for inconsistent information. Here the open

world features of OWL are implemented to provide a pragmatic solution to a modelling challenge (Dey, 2001). These examples demonstrate that the ontology driven approach present different behaviour in data management and this behaviour could be potentially beneficial in the railway context.

One of the objectives of the research is to investigate if this concept can be applied to a large scale system scenario where data is integrated from multiple heterogeneous systems. The focal point of railway systems interoperation was the ability of the ontology to cater for heterogeneous data. That is to integrate data from multiple sources that effectively represent the same information but in different formats based on differing model structures. This was the stumbling block discovered in the work on Engineers' Workbench where the models were neither sufficiently semantic nor extensible in solving the challenge (Elphick, 2004). The subject of open world assumption presented an opportunity to investigate this need by catering for inconsistent data. Addressing this matter requires the analysis of how OWL models are constrained. One of the hazards of attempting to represent heterogeneous information is that constraints can be created that can ultimately lead to inconsistencies in the model.

From an engineering perspective, the process of achieving this interoperability must be undertaken with input from, and interaction, with the stakeholder experts. While the open world situation can indeed be dealt with in closed world database systems, this process requires database experts to configure and model the theory that supports the necessary definitions. The concept of OWL ontology modelling was promoted as alternative to database modelling where the industry experts could participate in the

ontology modelling process (knowledge modelling). However, it was recognised that this activity in itself is not trivial. There are situations where a semantic model can in fact become inconsistent, i.e. the model classes and axioms represent a theory that cannot be satisfied. The modeller needs to be aware of this situation and requires training to avoid creating models which result in errors being thrown up during processing. The well-known pizza ontology examples provide a useful demonstration of a situation where a model cannot be satisfied and therefore demonstrates how the open world assumption can in fact be problematic. One of the key features of ontologies that are described using OWL-DL is that they can be processed by a reasoner. One of the main services offered by a reasoner is to test whether or not one class is a subclass of another class. By performing such tests on all of the classes in an ontology it is possible for a reasoner to compute the inferred ontology class hierarchy. Another standard service that is offered by reasoners is consistency checking. Based on the description (conditions) of a class the reasoner can check whether or not it is possible for the class to have any instances. A class is deemed to be inconsistent if it cannot possibly have any instances (Rector, 2008).

In the context of the pizza example, pizzas are defined as all having tomato and cheese toppings. A rule is recorded that states that all named pizzas must have at least one extra topping. Pizzas are classified as either vegetarian or meat pizzas based on whether they have some vegetable topping or some meat topping. In a closed world system a pizza with both vegetable and meat toppings throws and error because it has no rule stating that vegetable and meat toppings are the same – therefore it assumes they are not. An open world system will decide that a pizza with both vegetable and meat toppings is both a vegetarian and a meat pizza. This is because there is no rule to

state that the toppings are not the same and therefore it infers that they are. The major difference in an open world system is that in order to get the correct classification, the world must be closed to the extent that the inference can be made. In this pizza example the closing axiom would need to state that a vegetarian pizza must only have toppings of type vegetable toppings and must have at least one vegetable topping (to differentiate it from a standard pizza). In turn a meat pizza must only have toppings of type meat toppings and must have at least one meat topping. The final condition that must be in place is record the vegetable toppings and meat toppings are disjoint – i.e. they do not belong to the same class. These restrictions on the classification are referred to as closure axioms which utilise existential and universal restrictions available in OWL-DL. Adding these disjoint characteristics to knowledge base makes use of the Unique Name Assumption, i.e., that is the assumption that different names always refer to different entities in the world. CWA systems have UNA whereas OWA systems do not (Sequada, 2012). Therefore the definition of the ontology concepts and their relations must be carefully defined if any benefit is to be gained from the OWA.

Existential restrictions, also known as `someValuesFrom' restrictions, or `some' restrictions are denoted in DL- syntax using□ the symbol □ i.e., a backwards facing E. Existential restrictions describe the set of individuals that have at least one specific kind of relationship to individuals that are members of a specific class.

Universal restrictions are also known as `allValuesFrom' restrictions, or `only' restrictions since they constrain the filler for a given property to a specific class. Universal restrictions are given the symbol □ □ i.e., an upside down A. Universal

restrictions describe the set of individuals that, for a given property, only have relationships to other individuals that are members of a specific class.

A common `pattern' is to combine existential and universal restrictions in class definitions for a given property. For example the following two restrictions might be used together, hasTopping MozzarellaTopping, and also, hasTopping MozzarellaTopping. This describes the set of individuals that have at least one hasTopping relationship to an individual from the class MozzarellaTopping, and only hasTopping relationships to individuals from the class MozzarellaTopping.

In addition to these restrictions there is the hasValue restriction which relates a class to specific instance of another class and cardinality restriction which specifies the exact number of relationships that an individual must participate in.

Knowledge based reasoners utilise these restrictions embedded into the OWL ontology model to infer explicit information from implied content. The reasoner function deals with the open world and closed world features of the ontology by using the classify the concepts based on the degree to which a concept is closed.

The research undertaken focussed on the development of ontologies that can cater for the variation in information and the context that the information is used in for the interest of the end users. The aim was to test how the reasoner differentiates one situation from another for the benefit of the end user. The publication generated during this research addresses this (Lewis & Roberts, 2010).

The implementation of OWA is a necessity for a knowledge base to operate in this way. A system based on CWA, such as a relational database management system, cannot produce a similar result because it does not share the facility to reason over its content.

OWA forms a key part of two main implementations for integrating data in a railway environment. The first is the integration of data from sources that are similar, but different, such as train models. For example, all diesel trains have at least one diesel engine, but some have just one engine at the front while others have multiple engines under each carriage. A fault developed in a train of one type can lead to quite different consequences to a similar fault in a train of another type, i.e. a train with multiple engines can tolerate an engine failure and keep moving, whereas a train with a single engine cannot tolerate a single engine failure•. This example demonstrates that while a train concept has many different attributes, individual train types have different features which rely on concepts in the knowledge base to store related information. Any information system that integrates data from different train types must be capable of reconciling these differences in the information model.

The second implementation of OWA covers the capture of context, such as tacit, circumstantial information held by domain experts. The idea of a context based approach has seen increased interest from the research community largely attributed to the advancement in applications supported by the World Wide Web. There are an abundance of examples of application in the real railway environment and as the complexity of the scenario increases, so the potential level of benefit gained from an OWA based system increases. The ontology pattern shown in Figure 3.4 supports the

capture of implicit information, which is 'reasoned' over to infer some explicit content. The important feature of this pattern is that there is some 'statusReliesOn' property between selected 'Status' concepts which aims at representing some abstraction of the real world condition of some real world object. The 'statusReliesOn' property is equivalent to the *O*(*b*, *c*) concept from the "blocks world" example, whereas the 'status' property is equivalent to *G*(*b*). In the general sense, building a model in this way enables the user to posit queries such as: is there a *system* with a status *S1* that relies on a system *y* that has a status *S2*?

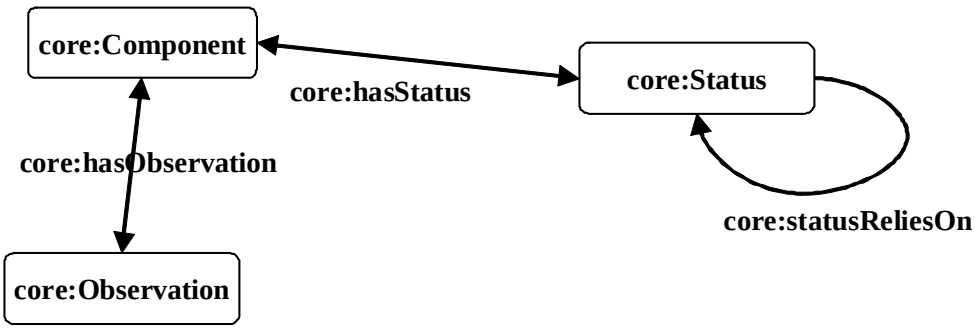


Figure 3.4 Measurement design pattern

A 'Symptom' concept captures domain knowledge from the experts such that a measurement with a certain level will infer that some real world symptom exists. In the simplest case, a *hot symptom* is inferred by a *symptom* associated with a *temperature* measurement with a *level* that is *high*. At this early stage and through following the design process, the ontology model is starting to reflect the natural language associated with a diagnostic process. Figure 3.5 illustrates the extension of the measurement pattern to represent relationships with 'Symptom', 'Fault' and 'Event Level' concepts. The 'Event Level' concept interprets quantitative data into qualitative information, i.e. it provides a semantic representation such as *high*, *low*,

hot or *cold*, etc. This is an important feature as it captures the terminology of the technical domain and begins to associate it with status information from the physical domain.

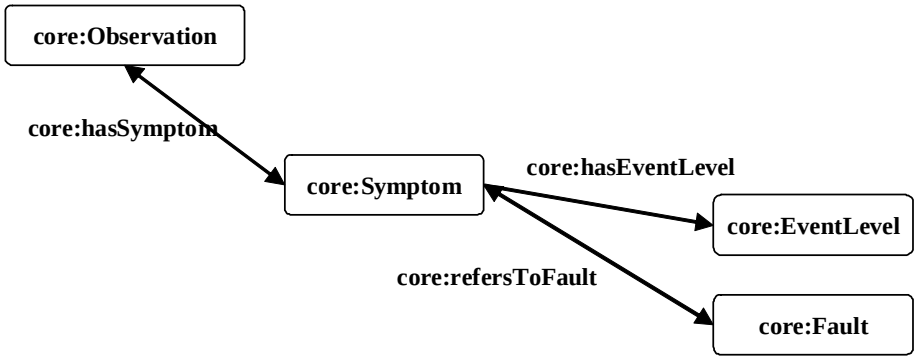


Figure 3.5 Extended measurement design pattern

The separation between the 'Symptom' and 'Fault' concepts caters for systems that provide raw data and those that perform detection and diagnosis. In the case where some symptom is inferred from some measurement data, a second inference is made to create an appropriate associated 'Fault'. This idea represents the basic mechanism for interpreting implicit information into explicit information for a domain based scenario.

3.3 Modelling the Railway Domain

3.3.1 Methodology

The subject of ontology design methodology is in itself an area of study for a number research schools leading the OWL development. Ontology re-use from other domains is a key influencing factor for new developments as there is no need to re-invent the ontology if one already exists. This suggests a modular approach to ontology design should be taken (Doran, 2006), The import capability of the RDFS standards allow ontology resources to be accessed over the Web, based on a known URL (Bechofer et al, 2004). This creates opportunities to share some common features and restrict others through local extension where there domain or application specific requirements. Ontology design for extension within the domain of development is also important. This promotes the hierarchical development of concepts from the highest, most generic concepts such that they can be extended for specific application. This is akin to software development where concepts referred to as design patterns form guidelines for software re-use (Gangemi, 2005).

The ontology design methodology promoted in this work implements ontology re-use through import and extension by the development of ontology design patterns. The methodology aims to create all generic models as patterns by non-domain experts with support from the domain experts, capturing all concepts that are considered necessary yet independent of the application. This step is relatively straight forward as the requirement is for an ontology of engineering components which means the variation is naturally limited. This leaves the domain experts to incrementally specialise the ontology to suit the requirements for interoperability in a large scale system with support from the ontology expert. This stage is more complicated as there

a more variable involved and more design decisions to be addressed. Therefore the ontology design is staggered such that interdependent blocks of development work are undertaken in an incremental methodology. This process commences with a scoping activity that answers questions about the scope and requirements of the information systems and applications that will produce, consume and share information in the domain. The methodology for railway domain ontology development is summarised in Table 3.1.

Table 3.1 Railway Domain Ontology (RDO) methodology

Step	**Definition**	**Example**
Define the Scope.	To define the engineering and information requirements of the domain. This includes component models which are required at the highest level that will be used in downstream application development.	The requirement to model infrastructure components and subcomponents such that measurement information can be attributed to the condition of those components.
Define the Ontology patterns for reuse and extension.	This includes the reuse of existing ontology resources for extension	The generic relations between tracks, routes and assets that relate to those routes.
Design the class hiercharchy of the core ontology concepts.	This is the first draft of the OWL ontology with defined class names and hierarchies as agreed with domain experts.	A track is a network edge, a junction is a network node.
Define the properties the relate the classes.	This is first stage to capture the generic relationships between engineering components.	Road network edges start and end at road network nodes.
Define the generic restrictions that form the key identifiers of concepts.	The capture of the universal, existential and value definitions and that uniquely identify the concepts.	A track network edge can only start at a depot, junction or station.
Check the core ontology concepts for consistency.	Ensure that the definitions are sufficient to ensure correct classification of the desired hierarchy.	A subclass of route does not get inferred as a subclass of track.
Define the use case and application specific concepts.	This requires the import and extension of the core ontology concepts.	An application dealing track asset management requires a track ontology and asset condition data ontology.
Extend the class, property and restriction concepts.	Requires additional information that domain experts require to differentiate one individual from another based on the necessary level resolution.	A track with a defect measurement has a restricted traffic speed.
Create instance data	Instance data is entered into the triple store to test the correct inferences are made.	Route Section A with any Track with measurement of type B is a route with restricted speed.
Develop the semantic architecture	Implement the required middleware and applications to deploy the ontology services and test the import mechanisms etc.	Triple store X is contacted and queried over http.
Verify the models	Implement a reasoning architecture and demonstrate that the triple store supports effective information management.	Deploy case studies and evaluate the results achieved.

The practical development employed a process of trying to establish some domain model patterns with the input of the domain experts. This proved difficult as the domain experts had no knowledge or experience of ontology design and time was short. This activity was backed up with modelling for a number of specific applications which were under review as part of a wider research scope within the Integrail project (Umiliacchi, 2007). The result of these two activities was what became known as the Railway Domain Ontology (RDO). This model held the generic high level patterns of concepts in the railway system. This RDO was subsequently imported by and extended with all other application specific ontology developments. The experience of modelling highlighted that a more formal approach should be implemented in future activities. Similar efforts were identified in other domains such as ontology development for medical informatics (Rector, 2003). A number of software tools were also considered to support the capture of the ontology and Protégé was selected because it was free to use and aimed at collaborative working (Tudorache et al, 2008). This tool allowed the stakeholder to contribute to the ontology design without necessarily being ontology experts.

The first process to check the validity of the ontology was to perform a consistency check. This was achieved by linking a reasoner, such as PELLET, to Protégé. Consistency checking is a fundamental activity to check that the constraints and entailments are valid and that no contradictions occur from the inference of the terminology (Baclawski et al, 2002). The second step was populate the assertion box with sample individuals to test that they are classified to the expected classes. This is referred to as realisation and allows the verification that the ontology is behaving as expected (Sandnes et al, 2008). This step is then repeated for any application ontology

that imports and extends the core RDO. When the stakeholders are satisfied that the ontology represents the intended meaning the ontology is ready for deployment in a Web application – possibly using the middleware and applications discussed in Chapter 2.

The final stage in the ontology development process is to verify that the models achieve their required objectives through some evaluation process. The latter chapters of this thesis describe how the ontology models were implemented as part of demonstration scenarios. The method for verifying the models and evaluating the results is described in the relevant sections. The chief criteria for evaluating the designs is to answer the following questions:

- Does the ontology import and extension mechanism provide sufficient demonstration of extensibility?
- Does the ontology design enable explicit information to be derived from implicit content?
- Does the ontology process support the functionality to integrate data from multiple different sources?
- Does the ontology capture sufficient tacit information to demonstrate potential for useful decision support applications?

The verification of the individual model implementations will be addressed in the relevant sections of the thesis.

3.3.2 Railway Domain Ontology Design

The process of ontology design for the application of decision support starts with an understanding of the information required by the user. For a large multidisciplinary system, this involves the collection of many requirements. These requirements represent general terms for the creation of general use cases. The principle behind the implementation of use cases is based on theory that the way the user interacts with the system is the key to the development of the ontology (Rector, 2008). The type of information required by the user guides the representation of relationships between classes in the ontology.

The development of ontologies is a hands-on process requiring the collaboration of many parties, including experts, within an organization. One of the problems of the activity was the potential flexibility in ontology development (Noy et al, 2001). This is because the structure of the ontology model is unrestricted. The application of ontology design patterns to represent classes and their relationships aimed at addressing this problem by generating a modular design process where patterns are re-used to avoid duplication of effort. The models defined in this case study were inspired by existing work in ontology design.

A small case study created to represent the integration of rail vehicle measurements with system data describes the features of the ontology design. It was based on the requirement of the rolling stock (RS) operator, the RS maintainer, the infrastructure (infra) operator and the infra maintainer. Each of these parties has an interest in the status of the vehicle and its potential impact on the infrastructure.

3.3.3 Railway Case Study

Wheel impact load measurement (WILM) systems describe a type of remote condition monitoring (RCM) system that detects the condition of the railway wheelsets from the trackside. These systems detect the forces of the individual wheel, the individual axle and the combined load of the vehicle. Hitherto, these systems identified vehicles with high wheel or axle forces, tagging them in an IT system for inspection and/or later maintenance. The integration of this data was usually limited to the attachment of vehicle identification data, either automatically through vehicle Radio Frequency Identifier (RFI) tagging, or manually through time/date and route data mapping from a scheduling database.

In the process of fault detection and diagnosis of an asset, the combination of data of different types and from different sources is vital to establishing an accurate depiction of the overall situation or *context*. In the case of railway vehicle wheel impact and loading data (WILM), the combination of that measurement data with other data, such as the train type and route information, enables the creation of a context for information. This context supports the querying of knowledge about the system status for different stakeholders. For example, the rolling stock operator wants to assess which vehicles are most in need of wheelset maintenance to avoid incurring charges from the infrastructure operator. The rolling stock maintainer wants to allocate resources while planning and prioritizing maintenance tasks. The infrastructure operator wants to make judgments of likely track condition of a route, or even a region, for life cycle costing (LCC) activities. The infra maintainer wants to make an estimate of the amount of force (dosage), that a section of track has been subjected to over a period of time for maintenance prioritization and resource allocation. In

addition, an estimate of the effect of the vehicle condition on other measured assets is required. For example, to establish the effect on a point machine alignment and subsequent operation by the repeated passage of trains with wheel defects.

Each measurement system is linked to a location in the physical system, which is usually represented as a text name. The key information, which is enough to identify the basic "train with fault" condition, is the measurement, the asset (train) and the location point. However, to support the information requirements described above, more context based system information is necessary. Fuchs et al, describe an application where a model layer represents the context of the domain and the instance layer (or knowledge base) represents the state of the world with respect to this context. This means that knowledge abstracted from the actual sources is expressed in terms of context information (Fuchs et al, 2005).

Figure 3.6 shows the pattern created to represent the location information for infrastructure assets. It shows the features of the railway infrastructure and indicates the differences between linear classes, i.e. 'Edges', and location classes, i.e. 'Nodes'.

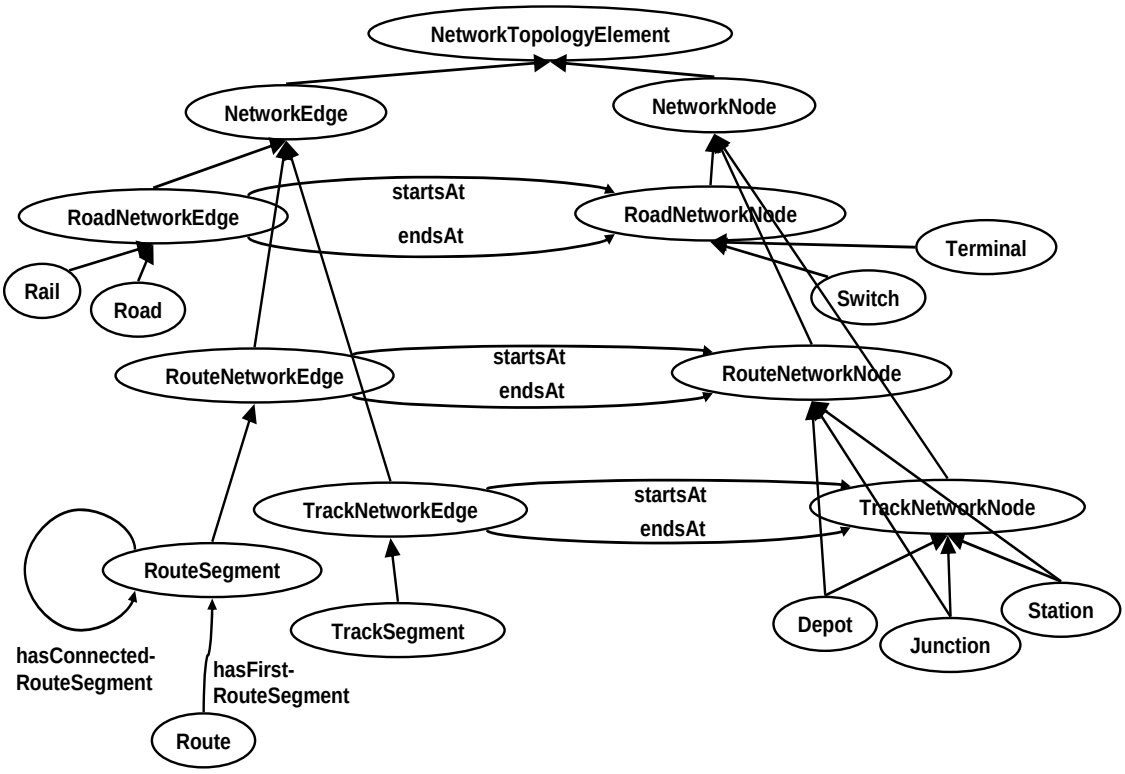


Figure 3.6 Transportation network pattern

This pattern provides a common structure for representing infrastructure features identified as important for querying by the domain expert. The patterns support querying by creating specifications for a 'RoadNetworkEdge', 'TrackNetworkEdge' or 'RouteNetworkEdge' and what represents the end points of these classes, i.e. a 'Switch' or 'Terminal' for a 'RoadNetworkEdge' and a 'Depot', 'Junction' or 'Station' for a 'RouteNetworkEdge' and 'TrackNetworkEdge'.

The completion of the pattern design enables the modeller to move to the next phase of the knowledge modelling process which is the specialisation of the pattern. This activity is specific to the application that is under development. In the case of the rolling stock operator, the focus is on the vehicle and its position relative to the infrastructure, whereas in the case of the infrastructure operator the focus is on the infrastructure. The application of the pattern design leads to the creation of instances of classes that occur in the real world, which is a similar approach to database design.

However, the ontology approach enables the convenient addition of more specific features, or *specialisation*, of those classes. For example, a 'Road' can be specified as starting and ending only at some 'RoadNetworkNode' and only consisting of 'Rails' and, even more specifically, only two 'Rails'.

Figure 3.7 represents coded Description Logic (DL) statements, a formal foundation for defining ontologies, applied to the ontology enabling this restriction. In this description, *ratom* refers to the property to be restricted and *catom* refers to the concept related to the property, *All* represents only the relationships that can exist for *Road* (note that in computer science the DL for the *All* restriction is described as the *Universal Qualifier*). The *atleast* and *atmost* terms represent the cardinality restrictions of the *isComprisedOf* property.

```
<catom name="Road"/>
  <and>
    <all>
      <ratom name="isComprisedOf"/>
      <catom name="Rail"/>
    </all>
    <and>
      <atleast num="2">
        <ratom name="isComprisedOf"/>
        <top/>
      </atleast>
      <atmost num="2">
        <ratom name="isComprisedOf"/>
        <top/>
      </atmost>
    </and>
  </and>….
```

This means:

'Road' *isComprisedOf* only 'Rail' and *isComprisedOf* at least, but no more than 2 'Rail'

Figure 3.7 Restriction Property for Road Concept

Two interesting features of the transportation network pattern are the 'RouteSegment' and 'TrackSegment' classes. The 'RouteSegment' represents the edge between two named points on the network, typically the stations. The 'RouteSegment 'is related to

another class called a 'Route', which itself has a property *hasFirstRouteSegment*, but also a *hasConnectedRouteSegment*which is a "transitive" property, as shown in Figure 3.8. The "transitive" property in this case means that any class related to one route segment is also related to all the other route segments for that route. This approach enables the definition of real routes within the ontology model.

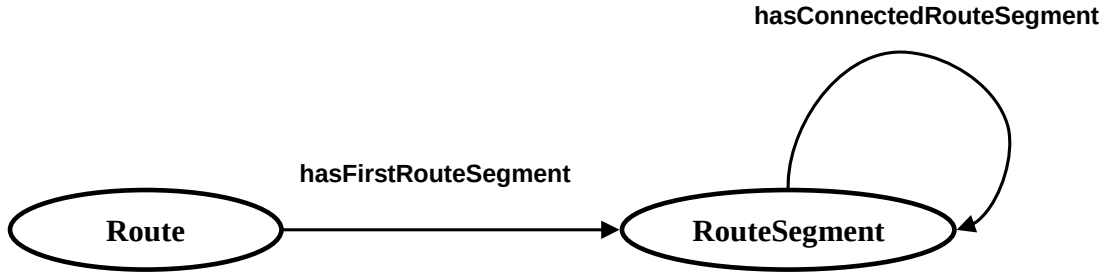


Figure 3.8 Route/Route Segment Pattern

Finally, a property established between the 'RouteSegment' class and the 'TrackSegment' class enables the differentiation between the operational and organisational view of the infrastructure. This is because the train operations are associated with route features, which is an abstracted notion of the track, while other aspects, such as measurements and maintenance activities, relate to the physical track. Track sections are related to position and location classes, which are represented as either global positions such as GPS locations or reference positions such as an Engineers' Line Reference[1] (ELR).

[1] All railway routes are measured from some datum point, often a convenient major place (for example, London termini). This leads to numerous routes with the same mileage, giving possible cause for confusion. To identify which route any given mileage is on, a three letter (sometimes also with a numeral) code is allocated to each line: the ELR. The code prefixed the structure numbers, giving each one a unique reference.

3.4 Modelling Observations and Context

The application of context is important in applications where an understanding of a wider perspective of the domain is beneficial to the design (Dey, 2001). Improving access to context makes it possible to produce more useful computational services. In the railway domain this is a key issue as measurement data combined with context provides a more powerful resource to decision making than the simple measurements alone. The *Track* classes already have some related context because a 'TrackSegment' is related to a 'RouteSegment', which is related to a 'Route'. This means that not only can the user query about a 'TrackSegment', but also the 'Route' to which the 'TrackSegment' and 'RollingStock' is related. The capture of the network model forms the foundation for the rest of the work because all of the other classes relate in some way to this model, i.e. a *Vehicle* is related to a *Route*, a 'WILMsystem' is related to a 'TrackSegment' and a 'PointMachine' is related to a 'Node'. The importance of this representation of the network will become apparent in the querying processes described in later sections.

Modelling these features requires the use of a number of pre-existing and novel ontology design patterns. For example, models representing the hierarchical nature of the railway vehicle have been constructed using the pattern of the *subsumption* property (*is-a*) and the *part-of* property (Lammari and Matais, 2004). A novel measurement pattern defined to describe the details of the observations made on the system is shown in Figure 3.9. This illustrates that a measurement class can have a number of properties that represent the temporal, spatial and quality aspects. The 'ObservationQuality' class provides an important meta-data in a decision making process based on system measurements (Strong et al, 2007). The additional classes

represent the context of that observation concept. The *Symptom* class provides an indication of what the 'Observation' that has been taken actually represents in terms of diagnosis. For example, an individual wheel force above a certain threshold represents a 'HighWheelForce' symptom, whereas a pair of wheel forces above a certain threshold represents a 'HighAxleForce' symptom. In addition, the 'Symptom' class is associated with a 'Fault' class along a property called *refersToFault*. This means that a specified 'Fault' is related to a particular symptom, e.g. a 'WheelFlatFault' is related to a 'WheelForceSymptom'.

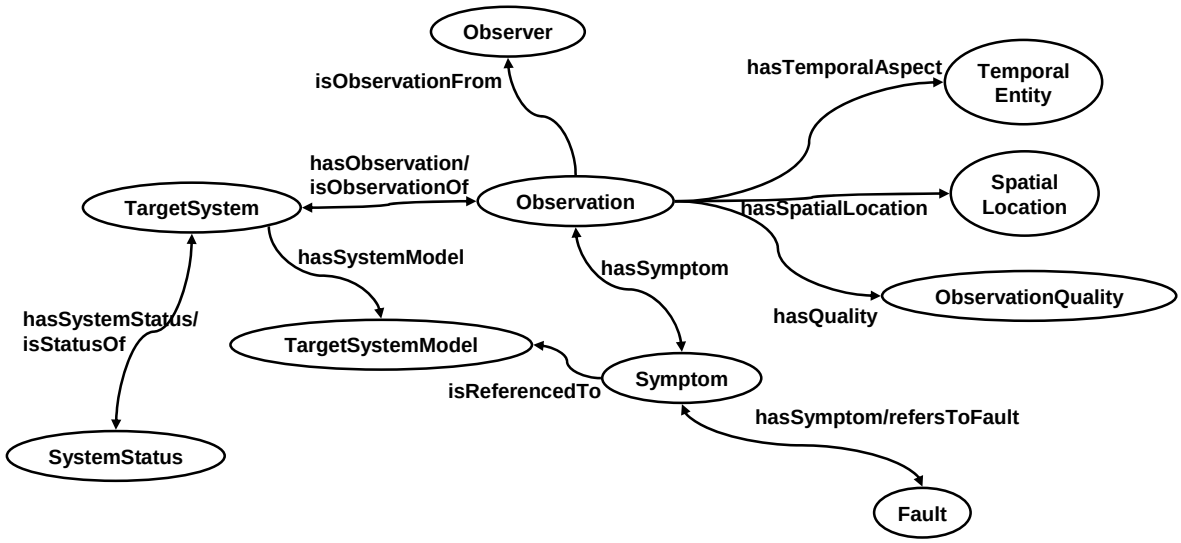


Figure 3.9 System Observation Design Pattern

A 'TargetSystem' is some part of the modelled physical system that is measured and therefore has an *Observation* class along the *hasObservation* property. A 'TargetSystem' also has a 'SystemStatus' class along the *hasSystemStatus* property, which implies that the status of a component cannot be known unless it has been observed.

The semantics that this design pattern records enables the representation of information that are more meaningful than measurements alone. They create the potential for an application that can "recognise" the information by classifying it according to the context model. The reasoner application introduced in Chapter 2 provides this classification making information implicitly represented explicit in the knowledge base and therefore creating new information.

3.5 Explicit Versus Implicit Information

Demonstrating the relationship between implicit and explicit information requires the translation of ideas developed in a conceptual manner into a realistic physical application. Two disciplines of railway operations are considered; traffic management and vehicle maintenance. The traffic manager wants to know if some system is faulty in order to decide what action to take; the maintainer requires the same information, specialised to support the required maintenance tasks.

From the traffic manager's viewpoint, the explicit information is that a train has an engine fault, wheel fault or similar, the implied information is that the train has a priority status and therefore should be removed from service at the earliest opportunity. In reality, this information is made explicit to the traffic manager through telephone calls, emails and other manual forms of communication. However, the exact detail of the fault is not important to the traffic manager, just that the train is a priority train and therefore needs to be dealt with appropriately.

The maintainer wants to know about faulty trains, but also wants to know the specific component which is faulty. However, querying on data to find the status of any

component can be cumbersome and time consuming, often involving some manual intervention to locate a faulty component. The maintainer does not want to have to rely on implicit knowledge of the system to locate the faulty component. To demonstrate this, a scenario was selected which integrates data from some infrastructure based measurement systems, measuring trains as they pass. These are wheel impact load measurement (WILM) systems and hot axle box detector (HABD) systems. A WILM system measures the force of each wheel on the track as the train passes. A HABD system checks the temperature of the axles and brakes (wheels) as the train passes. These systems represent important condition monitoring features, as a train with high wheel forces can cause cracks in the rail. A train with a hot axle can suggest an impending bearing seizure, leading to damage to the track and even derailment. Trains that are identified to be in this condition are removed from service as a priority.

3.5.1 Capturing Implicit Information

A domain expert may make a statement such as "the health of a train relies on the health of all of the vehicles, and each in turn relies on the health of the bogie, which depends on the health of the wheelsets", and so on. In a database application, this information is made explicit either through the maintainer's knowledge or through look up tables linking wheel data to vehicle data. Nothing is inferred from this data as it is entered and so applications are relied upon to derive information from the data. In an ontology based solution, a more general model is implemented that captures information implicit to the domain and uses it to produce new explicit information.

To support this idea, the concept of 'Status', introduced in the previous section, is specialised to 'PriorityStatus' and 'Non-PriorityStatus'. Explicit information about the dependency between components is captured using the *statusReliesOn* property. Figure 3.10 represents a model proposed to capture the dependency between components for a railway vehicle. In this diagram the "…" concepts indicate that there are other components that are depended upon i.e. that this model is extensible from the current version presented.

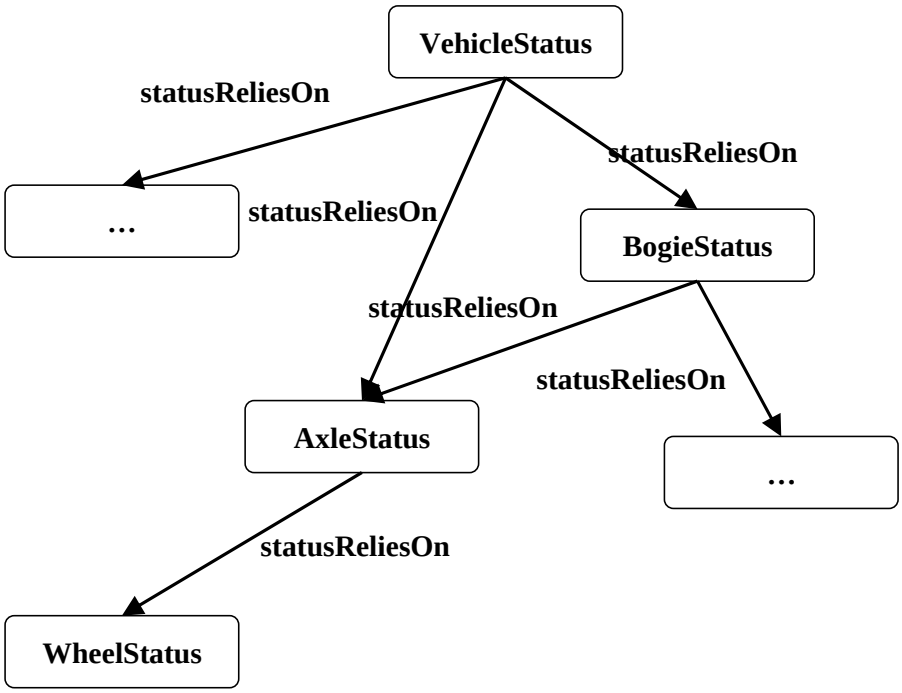


Figure 3.10 Explicit model for railway vehicle

The status of the vehicle is inferred by solving the problem of whether there is a non-priority concept that relies on a priority concept. The thinking behind this proposal is that the vehicle status is not updated directly and is therefore unknown until this problem is solved. The OWA feature means that this unknown aspect can be used in an inference process to entail that there is indeed a non-priority concept that relies on a priority concept. As a result of this feature, concepts that can be in one of two states

are considered to be in either until a fact is asserted that confirms the status to be *true* or *false*, therefore *closing the world* for that concept.

To describe the problem to be solved, it is proposed that the 'BogieStatus' is a *non-priority*, and the 'WheelStatus' is a *priority*. The knowledge of the system is represented as first order logic facts, where the concepts are *v*, *b*, *a*, *w*, representing Vehicle, Bogie, Axle and Wheel, with predicate symbols *R* and *P* representing the properties *statusReliesOn* and *Priority.*

{R(v, b), R(b, a), R(v, a) R(a, w), not P(b) and P(w)} - (7)

This represents the facts in the knowledge base, which are used to solve the problem represented as the general term:

P(x) and not P(y) and R(x, y) - (8)

If it is postulated that 'AxleStatus' is a *non-priority*, then the problem is solved by:

not P(a) and P(w) and R(a, w) - (9)

If it is proposed that 'AxleStatus' is a *priority*, then the problem is solved by:

not P(b) and P(a) and R(b, a) - (10)

In either case the problem is solved and the vehicle is found to have a priority status.

In scenario building this type of processing is very powerful. Expanding the model to

infer the train status enables the traffic manager to gather information on priority trains and assess its effect on the operation without needing to understand the exact fault. The maintainer would apply similar logical propositions to identify components as priorities for maintenance. In this case the requirements are more specific as the maintainer wants to know about more specific faults. In this case, the inference is aimed at priority bogies or priority wheelsets, etc. However, in both cases the model structure is the same, which illustrates the generic nature of the approach.

This example illustrates that the ontology provides a standard for information interchange that can be shared by the producer and consumer. The terms 'Vehicle' and 'VehicleStatus' form part of a vocabulary which is shared by applications; each application that references a copy of the ontology can infer meaning from the information in the same way. The ontology provides an extensible mechanism for recognising context in the information stored.

3.6 Railway Domain Ontology

An ontology resulting from requirements to integrate data for a system as large as the railway domain is necessarily large and potentially cumbersome to manage. The "Railway Domain Ontology" (RDO) resulting from the investigation into the railway industry needs for integration is based on many competing requirements. The design features most prominent in the working ontology were those that resulted from application level decisions. However, these features had to compete with the requirements to maintain the generic, re-usable features of a sound ontology approach. Therefore, the decision to maintain a structured approach resulted in delineation between the core ontology design and those features directly related to an application. While the applications influenced the features of the core ontology, the aim of the ontology itself was to remain as generic as possible. This approach is in keeping with the view of Uschold and Jasper who propose a framework for ontology design consisting of a set of ontology application scenarios (Uschold and Jasper, 1999). These scenarios broadly group into three application areas: to assist in communication between human agents, to achieve interoperability or to improve the process and/or quality of engineering software systems. The requirements for ontology application in the railway domain are grouped based on use-cases defined by industry stakeholders and from known integration requirements from consultation and publication, etc. The high level objective for the RDO is to provide fusion systems, combining information acquired from various sources, and applied to the various facets of the railway as presented in later chapters of this thesis (Kokar et al, 1995).

3.6.1 Ontology Mapping

One solution to the challenge of managing large and potentially cumbersome ontologies is to implement a method called a "reference architecture". The objective of such an approach is to create a dynamic environment where the ontology resources and the applications that use them are distributed. Such architecture is applicable to Semantic Web applications such as pervasive computing paradigm (Saha et al, 2002). Since there are a variety of ontology management tasks, the activity can be externalised (Lee and Goodwin, 2006). While this activity is in some way outside of the main scope of research, its application is so important to the deployment of the working system that some effort will be given to describing the main features of its use.

The reference architecture for the railway domain is based on the core ontology concepts. Therefore, any application which chooses to specialise a general class for this domain must make reference to this ontology. The core ontology itself makes reference to other generic online resources. The number of referenced resources will grow as the complexity of the application under development grows. In the case of a vehicle monitoring and maintenance scenario there are numerous resources, each one relating to a different feature of the application scope. Figure 3.11 shows the OWL model for this application where two reference mechanisms are applied. The first is the XML namespace protocol represented by the prefix "xmlns" which is used to uniquely identify elements of an XML document, of which OWL is a derivative. The second is the "owl:imports" annotation which signifies the Uniform Resource Identifiers (URI) of the ontology that the current ontology makes reference to. For the example given there are five ontology resources that are imported.

```
<rdf:RDF xmlns="http://www.integrail.info/ont/IMAINEventManager.owl#"
    xml:base="http://www.integrail.info/ont/IMAINEventManager.owl"
    xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
    xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#"
    xmlns:oc="http://www.integrail.info/ont/IMONObjectCondition.owl#"
    xmlns:habd="http://www.integrail.info/ont/HABD_1.owl#"
    xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:wilm="http://www.integrail.info/ont/WILM_1.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:sp3b="http://www.integrail.info/ont/sp3bOntology_v5#"
    xmlns:core="http://www.integrail.info/ont/SP3A.owl#">
   <owl:Ontology rdf:about="">
      <owl:imports rdf:resource="http://www.integrail.info/ont/WILM_1.owl"/>
      <owl:imports rdf:resource="http://www.integrail.info/ont/SP3A.owl"/>
      <owl:imports
rdf:resource="http://www.integrail.info/ont/IMONobjectcondition.owl"/>
      <owl:imports rdf:resource="http://www.integrail.info/ont/HABD_1.owl"/>
      <owl:imports
rdf:resource="http://www.integrail.info/ont/SP3BONTOLOGY_V5"/>
   </owl:Ontology>
```

Figure 3.11 Example of ontology referencing architecture

The ontology referencing approach is based on World Wide Web Uniform Resource Identifiers. This means that any modelling concept has its own URI prefix, as described in Chapter 2, that not only uniquely identifies it but also references it to an ontology resource to which it belongs. In the case of the core railway domain ontology the prefix is set to http://www.integrail.info/ont/SP3A.owl. The importance of this referencing mechanism is significant and important to understand the integration work described in this thesis. A simple *individual* or *instance* is created and referenced to a local ontology such as:

http://www.integrail.info/ont/IMAINEventManager.owl/data#Symptom_1.

Figure 3.12 illustrates the relationship between the component ontologies through a reference architecture. Even though the core ontology is not local to the application that this instance of symptom was created in, because of the reference to the core ontology, http://www.integrail.info/ont/SP3A.owl, this instance is referenced to that ontology through the URI link. It can therefore be stated that this works because the local ontology "knows about" the remote core ontology.

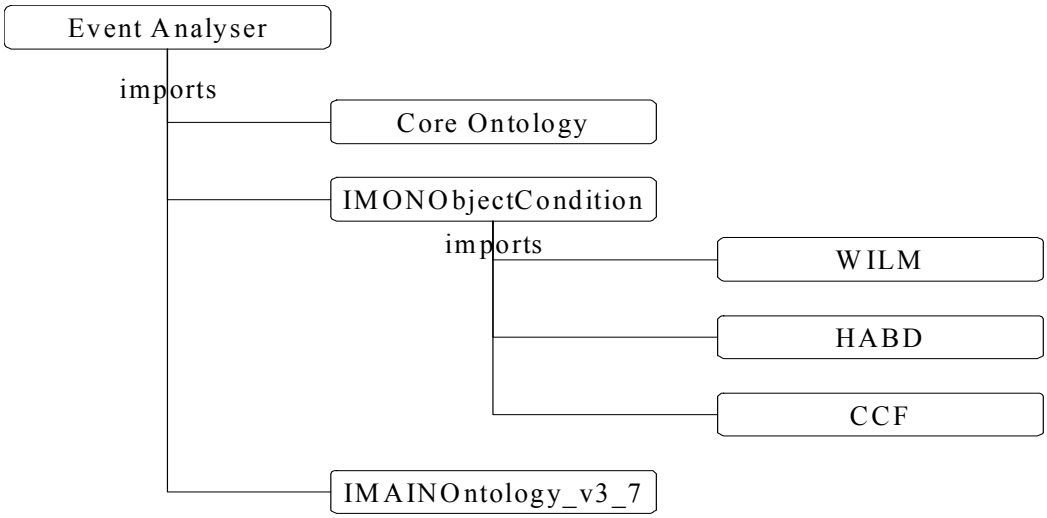


Figure 3.12 Ontology Reference Architecture

The relationship between a local symptom *Symptom_1* declared in http://www.integrail.info/ont/IMAINEventManager.owl/data and the remote *core* ontology is shown in Figure 3.13 where *Symptom_1* refers to an observation *WheelPeak_6* which has a high event level associated with it.

```
<core:Symptom rdf:ID="Symptom_1">
        <core:refersToObservation rdf:resource="#WheelPeak_6"/>
                <core:hasEventLevel
          rdf:resource="http://www.integrail.info/ont/SP3A.owl#high"/>
                <core:refersToFault>
         <core:Fault rdf:ID="Fault_1">
```

Figure 3.13 Relationship Between Local Symptom and Remote Core Ontology

3.6.2 OWL-DL representation

The description logic extension of OWL enables the representation of logical axioms of the type described in the previous example. Using the OWL standard introduced in Chapter 2, logic expressions such as that presented in (2) can be written in the form:

hasStatus ∃ (reliesOn ∃ (NonPriorityStatus ⊓ (hasStatus ∃ (reliesOn ∃ (PriorityStatus)))) -(11)

The format of the axiom described in (5) forms a pattern for inference of the status of systems monitored. The reasoner application uses this type of axiom to infer over the ontology content, identifying implicit status information from the explicit model. However, to enable this reasoning, a number of component applications are required.

Ontology design activities are supported through the application of an ontology editor tool. There are many of these tools available, some commercially, others as freeware. The selected editor for the work described here was Protégé, which is open source

freeware available from the Protégé developers (Gennari et al, 2003). Protégé was chosen because it has many design features that support the creation, visualisation and manipulation of ontologies.

3.7 Conclusion

This chapter introduced a set of design criteria for the delivery of ontology supporting applications in a particular domain. Ontology design is founded on the development of well understood patterns resulting from four design criteria:

- representing the sequential relationship between physical components;
- representing the abstract dependencies between them;
- representing their relationship with external observations;
- representing mappings to concrete data types, such as legacy data, for historical analysis.

This chapter has described an ontology modelling and design process that captures these criteria through the development of an OWL ontology model. Where necessary, the OWL standard is appended with description Logics (DL) to support the representation of logic axioms required to capture the domain knowledge. Later chapters will demonstrate how these design criteria are applied to support case studies for the deployment of knowledge based applications for the railway domain.

CHAPTER 4

TRAIN BASED DEPLOYMENT

4.1 Introduction

There are three functions for which an ontology based approach is applicable: (1) to assist in communication between human agents, (2) to achieve interoperability or (3) to improve the process and/or quality of engineering software systems. Each deployment of an ontology centred architecture may offer one, two or even all three of these features, with the exact nature of the deployment dependent on the specific requirements for information interchange and integration. Use cases were defined that aimed at demonstrating one key feature of potential benefit to the stakeholders.

The overlying objective was to overcome the limitation of earlier standards such as XML and XML schema investigated in previous railway integration research (Shingler and Umiliacchi, 2003). The difficulty faced was the ambiguity in defining conceptualisations and the relationships between them. Although XML has provided a single interchange format, different users can model the same data in different ways, which can lead to heterogeneities at various levels, including the semantic level. Semantic heterogeneities can arise from entities being perceived differently. For example, an agricultural expert perceives waterways to be a source of irrigation, while a transportation expert perceives them as a mode of transportation. Such perceptual differences manifest themselves as heterogeneities in the data models these experts

develop. These differences cannot be resolved without knowledge of the models of the underlying disciplines (Cruz and Rajendran, 2003). The reality of an implementation of XML using XML Schema for information interchange is that the ability to encode any semantics is limited to nesting between concepts and datatyping. The result of this is that the ambiguity cannot be resolved without a specific negotiation of the interface between the interoperating system. Furthermore, this negotiation must be repeated for each and every interface that joins the interaction.

True interoperability cannot be achieved with this piecemeal approach. The work on Network Rails' Engineers' Workbench demonstrated this as a single XML Schema was implemented as the interface to centralise data from all types of condition monitoring system. The objective was eventually achieved but not without manual intervention to encode the data into the schema format by negotiation with the data producers. The designers were exposed to low level implementation issues that XML Schema implementation was aiming to avoid. This problem was recognised and research was undertaken to develop conceptual, logical and physical level models in UML class diagrams to capture the semantics in the data (Routeledge et al, 2002). While this provides some control of the design mechanism to create an interface document that restricts the content of the XML instance data, there is still only a nested structure to capture the meaning of the terms. The computer interpretable content is not rich enough to truly avoid ambiguity. While this approach would have at least assisted the interoperability in the railway context – the potentially large numbers of different system presenting different data still posed a challenge that XML and XML schemas could not fully cater for. In contrast the ontology driven approach that was prevailing was promising truly unambiguous models operating over state of

the art Web service architectures (McIlraith, 2001). The aim was to separate the effort of the experts from the implementation of the information interchange mechanism.

Since the Web Ontology Language (OWL) standards promised to remove the ambiguity from XML interpretation, one of the key aims was to demonstrate how this could be achieved in a complex environment such as a railway system. A second aim was to address the challenge of interoperability or more specifically the handling of heterogeneous data types. Any data integration exercise in a system as large as the European railway network will generate a number of situations where it is necessary to interchange, integrate and process data that is similar in nature in terms of the information it represents, but differing in actual representation. The complexity of these situations cannot be estimated until the integration is attempted, which means that a large scale collaboration exercise is required involving a party of numerous and differing stakeholders. This is a recognised problem for ontology developers; the work of Pinto goes some way to establish some ground rules (Pinto, 2004).

4.2 Train Scenario Investigation

There are two areas of focus considered in this work: the first is the train operation which is closely tied with traffic management; the second is the infrastructure management. While the two are inherently linked, they are operated as two independent systems. The business drivers for each are very different as the focus of the train operators is on the passenger (or freight), whereas the focus of the infrastructure managers is on the train operator and other stakeholders such as governing, regulating bodies.

Railway companies, like other large industries, use Key Performance Indicators (KPIs) as a measure of success and a benchmarking focus for the operations of the business (Anderson, 2009). Putting the demonstration scenarios into the context of KPIs is an activity which ensures that the resulting ontology based system meets the fundamental criteria of the domain. Figure 4.1 shows a KPI tree resulting from the investigation of parameters affecting the management of traffic in the European rail network.

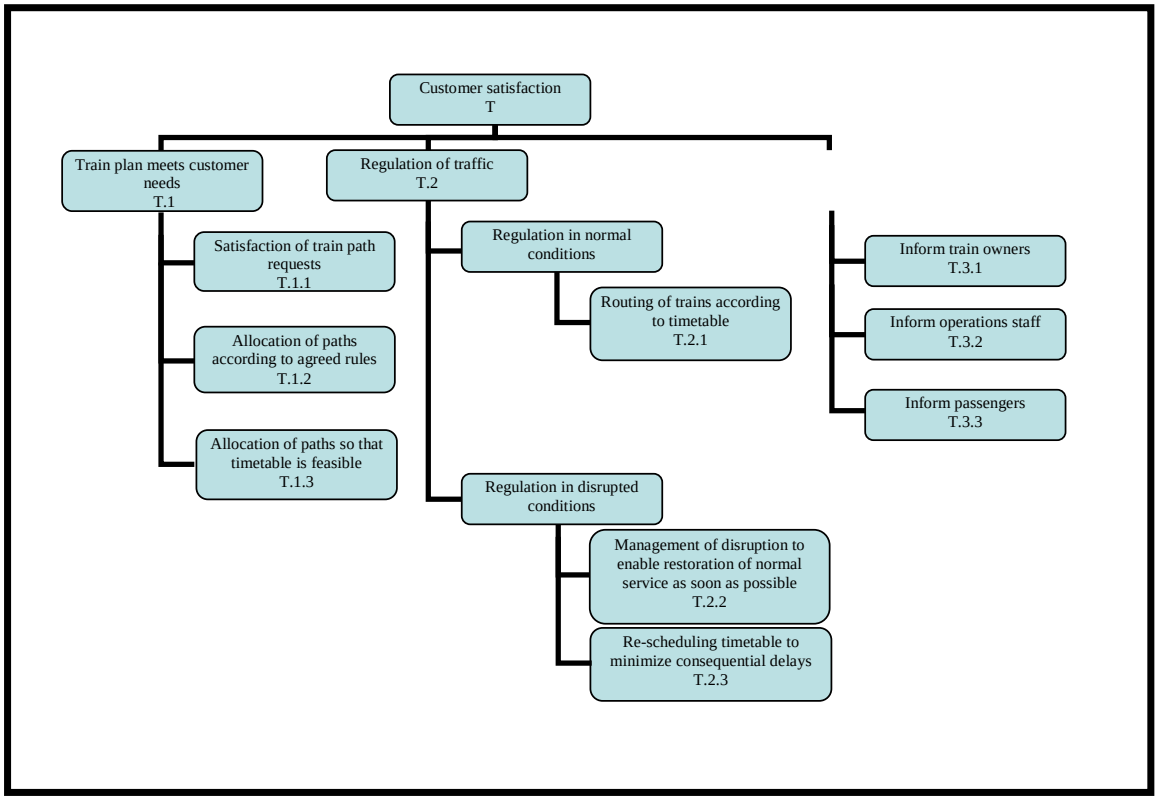


Figure 4.1 Extract from Traffic Management KPI

From an operational perspective, the elaboration of T.2 yields that the KPI most relevant to the type of demonstration required for the ontology based system is that of the management of disruption to enable restoration of normal service and T2.2 in

particular. The analysis of T2.2 resulted in a functional outline to provide incident management systems which will:

- implement strategy to reduce impact;
- support operational procedures to minimise hazards;
- automatically execute actions to minimise faults and hazards.

From a vehicle management perspective there are a number of KPI functions which can be utilised to assess the performance of the ontology based approach. Figure 4.2 illustrates the KPIs for the vehicle where the analysis yields a focus on 'Check and Monitoring of System'.

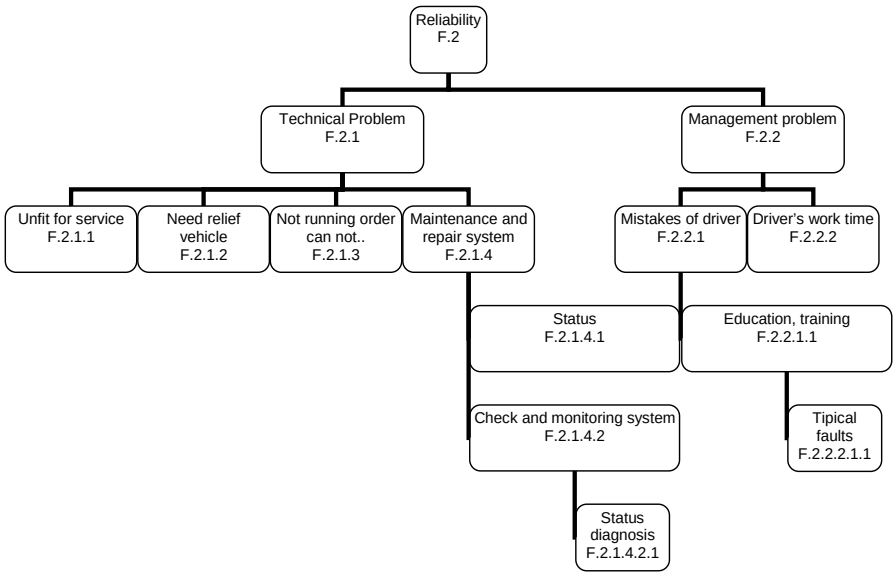


Figure 4.2 Extract from the Vehicle KPI tree

Demonstrating that the KPIs of traffic management can be met required the investigation of a current real world scenario. In the case of UK operations, if a train develops a fault while in service the majority of communication resulting from an event is manual, based on telephone calls. The train models considered in this

scenario are DMUs (Diesel Multiple Units[†]) which are managed as a 'consist' (a collection of DMUs) by a networked system called a TCMS (Train Control and Management System). The main function of the TCMS system is to control the various features of the train, such as doors, and interfacing to subsystems such as the engine management systems, etc. However, the TCMS also records data associated with events that are outside of the normal operating parameters of the system. This data is typically analysed subsequent to an event to support recovery activities and the management of disruption.

The demonstration chosen resulted from discussion with personnel at Alstom who are responsible for the manufacture, servicing and maintenance of a number of train fleets operated by a number of Train Operating Companies (TOC). Figure 4.3 illustrates the use case for the current scenario where a collection of actors collaborate to deal with an event, such as the discovery of a TCMS fault. When the fault occurs it is displayed as a syntactic code on the train management driver display which the driver calls/radios through to the signaller or helpdesk (Operator). The signaller/helpdesk uses the information to consult with fleet planning and make decisions about the impact of TCMS events on planning, on service and on maintenance (Consult). In the case where an intervention is deemed necessary, the arbitrator from the infrastructure side collaborates to apportion the cause of failure (Apportionment of Cause). The role of System Support is driven through the daily conferences organised and run by the Fleet Planning (Confer), and through the use of RailSys to raise a 'Service Order' for carrying out the required work (RaiseServiceOrder). At present the Fleet Console is used as a supporting tool in this process. The overall goal of this work is to make the

[†] Where each vehicle possesses a diesel engine beneath the carriage

Fleet Console the driving factor, underpinning all decisions and processes associated with event management. Figure 4.3 illustrates a use case representation of the current approach to fault handling.

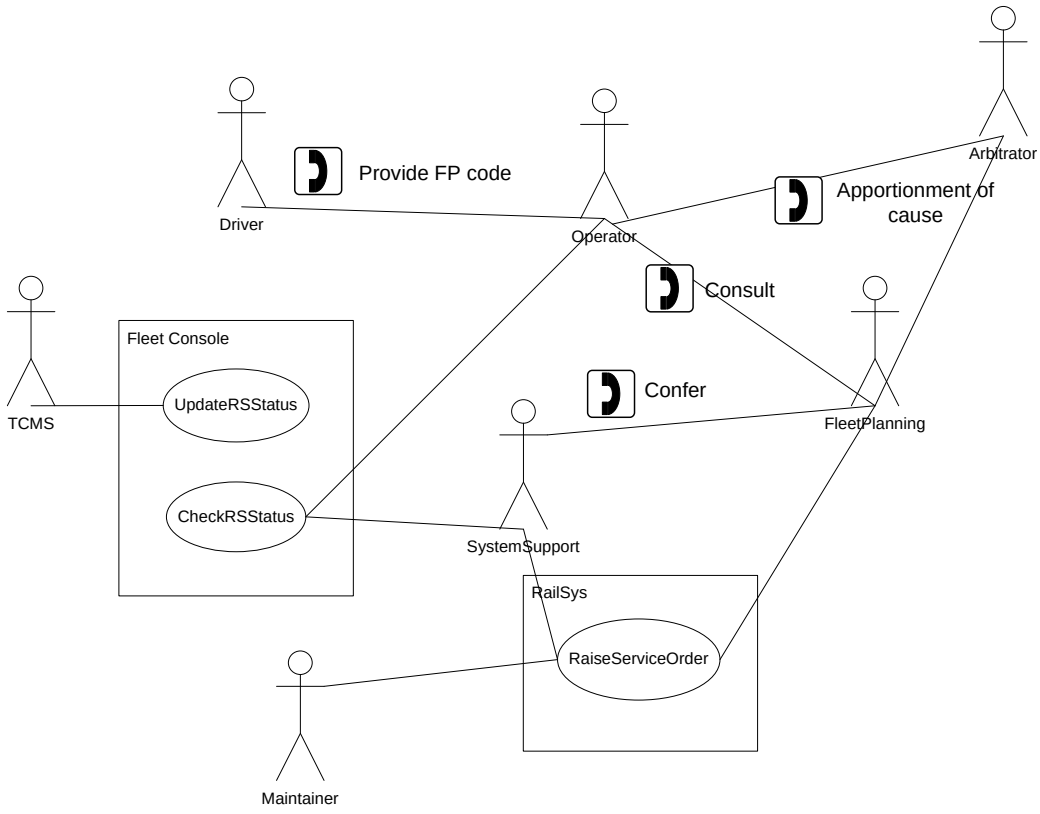


Figure 4.3 Current Scenario for fault handling

The first part of the proposed approach focuses on creating a semantic model of the information stored in the syntactic fault code set along with the capture of the tacit knowledge of the signaller/helpdesk (Operator). This model supports the automation of the process supporting decisions underlying the fault management. Figure 4.4 illustrates how the Fleet Console is appended with functionality to automate part of the consultation and conference process between system actors. The sequence of this diagram is initiated by an event generated by the TCMS system. The proposed system will automatically generate work orders and, where necessary, produce

appropriate event indications. The goal is to infer relevant scenario data such as allowable train speed, maintenance requirements and planning to handle the rolling stock, and ultimately provide a tailored view of this information for all actors involved.

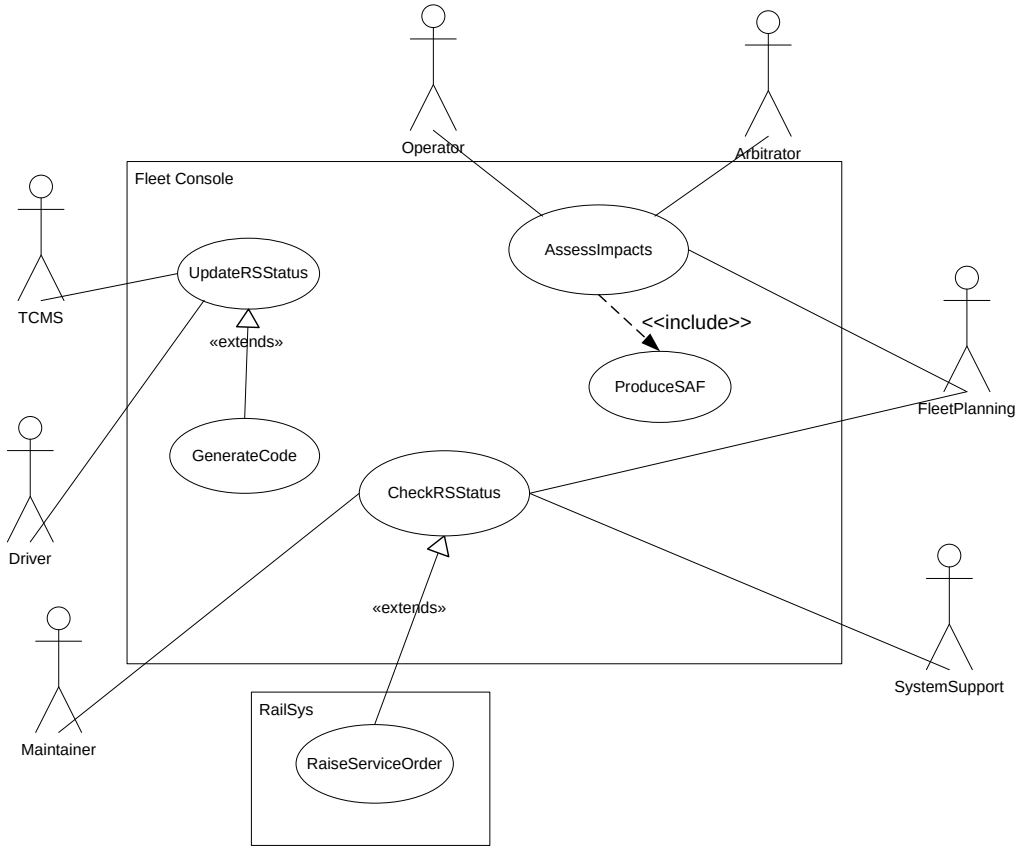


Figure 4.4 Scenario for enhanced support system

In undertaking this work the following tasks were proposed:

- A detailed use case analysis covering current status and indicating areas for improved benefit for the customer;
- Create a semantic model from syntactic representation of fault codes;

- Mapping of TCMS fault codes to fault codes held in the semantic knowledge base;
- Create automatic inference of effects of events for main actors – perform interviews with domain experts as supporting activity;
- Demonstrate an improvement in the arbitration process through automatic knowledge driven system interactions;
- Demonstrate benefit to the business process through improvements to KPIs and reliability;
- Prove scope to automate process from 'System Support' layer to 'Railsys' to 'Fleet Planner'.

The result of this work is the demonstrate the potential of a reliable knowledge based autonomous system for event management and decision support.

4.3 Analysis

Prior to consideration of the specific knowledge base features of the ontology based systems, there is a need to satisfy more general software design criteria in a comprehensive software design cycle. Figure 4.5 presents a high level use case diagram for the system that supports the proposed scenario. The use cases relate the actions of configuring the prototype system in a realistic manner so that remote resources can make use of the available functions in a later, more elaborate version of the system. The use case diagram presents three system actors, *Status Updater*, that registers a monitoring system and uploads data from it, *Status Consumer*, that subscribes to status notifications and then acts on them accordingly by identifying the assets affected and passing calls to the reasoning service and *Event Consumer*, that

subscribes to event notifications such as the results of reasoning services and subsequently acts by discovering associated maintenance and operational requirements.

The features of the use cases are generic to support application to a number of deployment scenarios. While a detailed development is required for a final system delivery, this is outside the scope of interest of this thesis, therefore a summary of the design process is provided in this section.

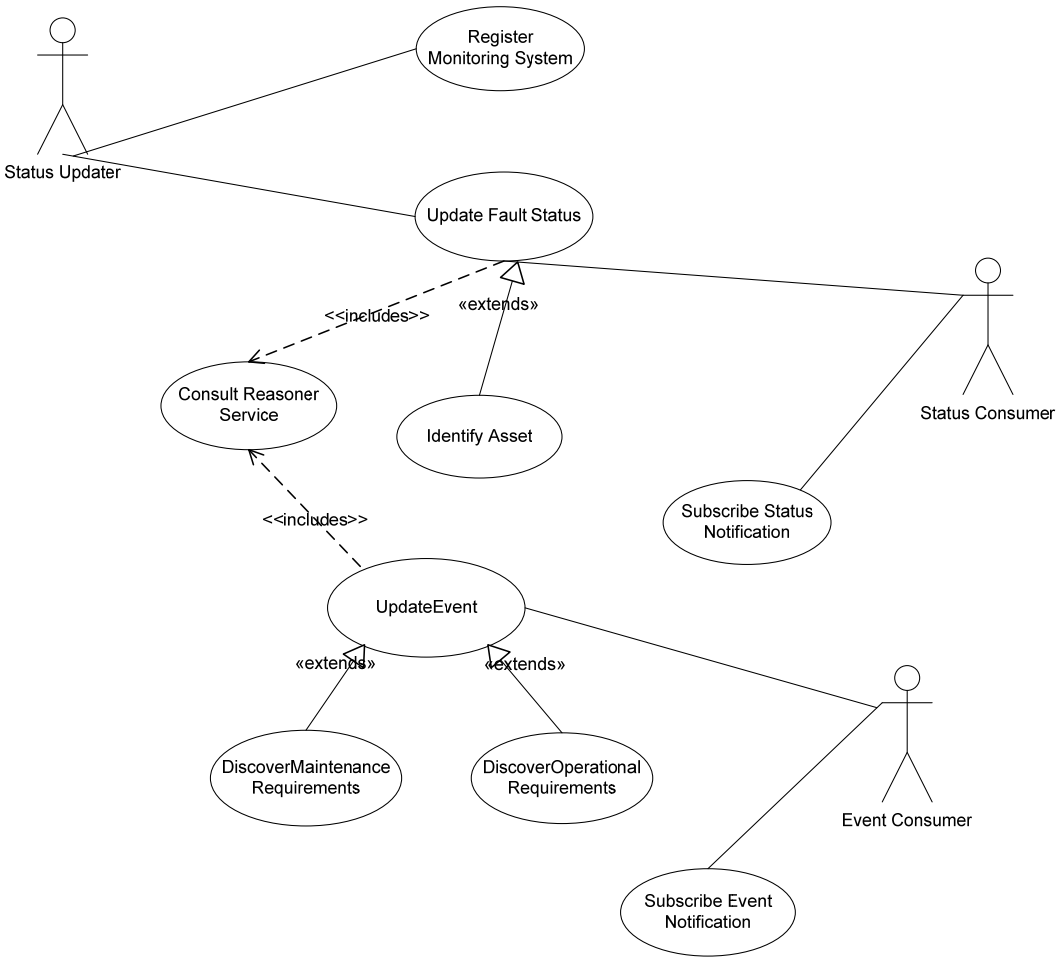


Figure 4.5 High Level Use Case Scenario Specification Diagram

In the context of the European railways there are many *types* of train fleet that are likely to operate over different *types* of infrastructure in possibly different countries and therefore there is a need for an unambiguous messaging service. The specific scenario considered in this work if that of a fleet of operational trains that contain an on-board rule reasoner for system diagnostics and an ontology based message transfer service. Therefore the aim of the scenario is to demonstrate two of three benefits of deploying a knowledge driven system, i.e. assist in communication between human agents and achieve interoperability. It is difficult to prove the third benefit of improving the process and/or quality of engineering software systems in this case as the demonstration is not broad enough to cover an equivalent system development. However, there are other examples of research activity that support an argument for a model based system such as that proposed in the standard ISO15926 (ISO15926). This standard represents a mechanism for integrating the life cycle data of process plants, including oil and gas production facilities. This includes a generic data model and a reference data library for process plants. The aim is to support the recording of data throughout the life cycle through the implementation of a structured modelling approach.

A demonstration is defined with respect to a well understood subsystem, the external train door, the behaviour of which has been studied and resulted in the analysis of failure modes (Lehrasab, et al, 2002). The demonstration considers the relationship between a pneumatic door actuator and the compressor system powering the door system described by these failure modes. The proposed solution requires the consideration of a simple fault where a door exhibits some symptom relating to a faulty status. The modelling process is simplified for the sake of demonstration,

though a more elaborate case would likely follow a similar design process, where the ontology design is based on a meta-level structure and implemented to create some contextual model (Noy and McGuinness, 2001), (Fuchs et al, 2005).

In practice, the requirements of the train fleet operator will differ from the train maintainer. The train fleet planner is interested in trains that appear to be failing and therefore necessitate management, whereas the train maintainer needs to know about the preventative maintenance actions required; the difference in these viewpoints also represents the *context* of the scenario.

4.4 Architecture

The architecture devised for the demonstration was configured using the Jena semantic web framework for Java. Jena provides an Application Programming Interface (API) which enables the creation of a repository for storing a model (T-Box) and instance data (A-Box) (Carroll et al, 2003). One of the main reasons for building an ontology-based application is to use a reasoner to derive additional truths about the concepts that are modelled. Jena includes support for a variety of reasoners. A common feature of Jena reasoners is that they create a new RDF model which contains triples that are derived from reasoning, as well as the triples that were asserted in the base model. Figure 4.6 represents this arrangement where the asserted statements, which may have been read in from an ontology document, are held in the base graph. The reasoner, or inference engine, can use the contents of the base graph and the semantic rules of the language to show a more complete set of statements.

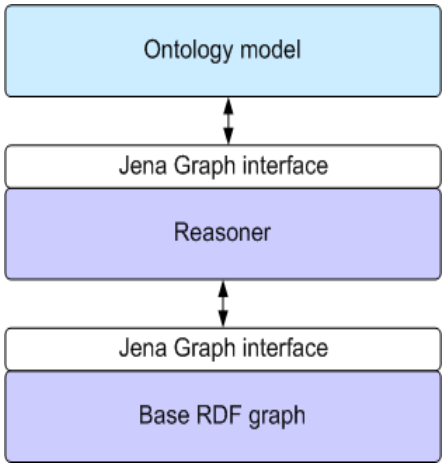


Figure 4.6 Arrangement of models in Jena

A prototype design for the rolling stock system consists of an OWL ontology model and a semantic reasoner configured to accept door symptoms and the symptoms of associated systems. The train controller system contains an OWL ontology specialised to capture the context of the railway operational characteristics. The train maintainer system contains an OWL ontology specialised to capture the context of the railway maintenance requirements. The controller and maintainer systems pose their own rule sets configured to identify particular characteristics in the repository data. This arrangement is deployed across three levels of asset, controller and stakeholder systems, as shown in Figure 4.7.

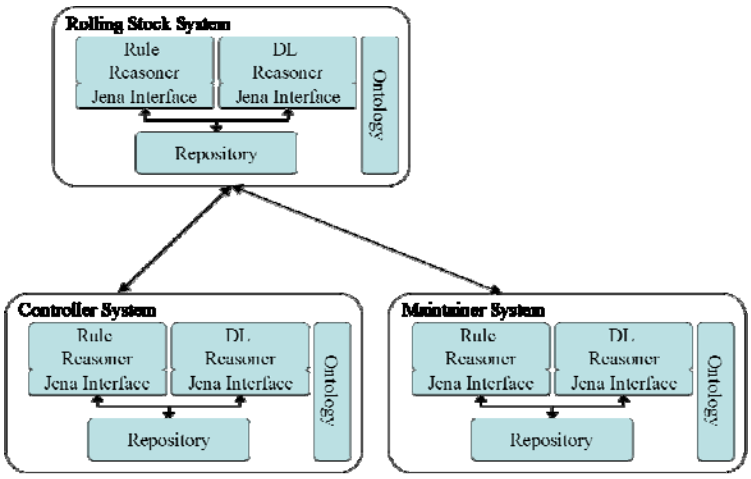
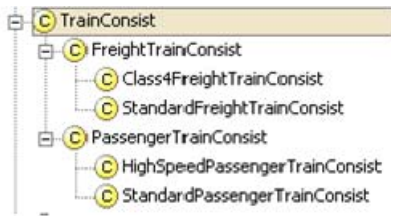


Figure 4.7 Scenario for Context Recognition

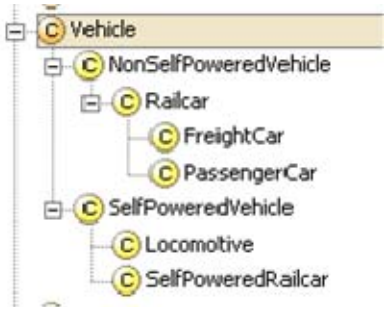
The communication between the component systems is based on a pull mechanism where the SPARQL (a query language) forms the interface standard (SPARQL, 2007). However, the mechanism in the proposed final system is based on a push mechanism. This requires an arrangement where the requestor subscribes to a service from a provider, which responds as and when relevant events occur.

4.5 Ontology

The ontology model supporting the demonstration is based on the mereologic, topological and taxonomic principles used in many ontologies (Borst et al, 1997). The ontology is based on *is-a* models of train and vehicle types and *part-of* associations between the vehicle and vehicle components. Figure 4.8 represents a graphical depiction of an *is-a* model for a) Train Consist types and b) Vehicle types. Figure 4.9 illustrates a UML representation of the *part-of* relationship between components where a single component can possess a hasPart relationship with many other components.



a)



b)

Figure 4.8 *Is-a* Ontology for a) Train Consist and b) Vehicle

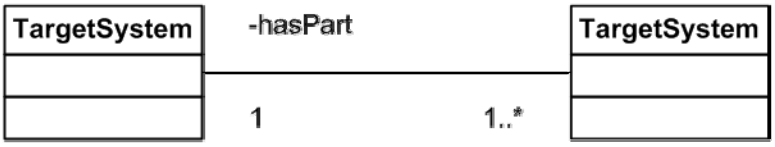


Figure 4.9 *Part-of* Ontology Concepts

A third ontology component represents the dependency between systems. This relationship is defined by creating a status characteristic for each component and then associating these components through the *statusReliesOn* property, as shown in Figure 4.10.

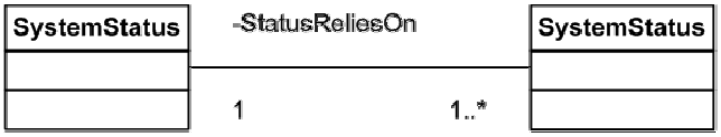


Figure 4.10 System Dependency Based On s*tatusrelieson* Property

The relationship between a component and a number of measurements that are taken of them is also represented, as shown in Figure 4.11 The observation model (generalised to accept input from human observers and monitoring systems) ties the system 'Component' to some value-timestamp pair, one or more 'Symptom' and ultimately one or more 'Fault' concepts.

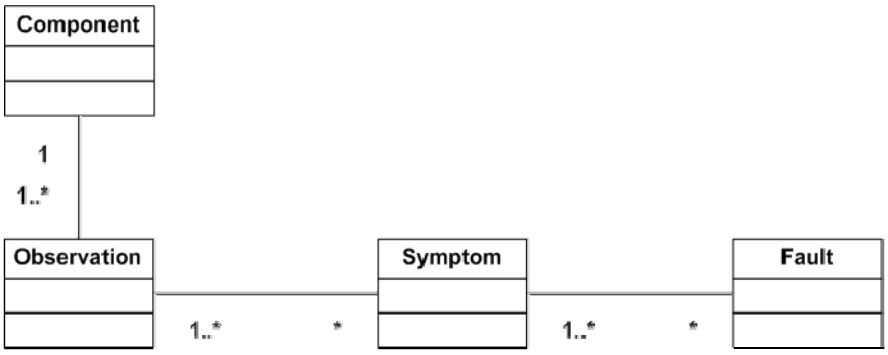


Figure 4.11 Ontology Measurement Concepts

4.6 Reasoning

In the deployed scenario, each train can make an assessment of its health status based on what is "known" about its components. The ontology model contains assertions about the relationship between observations, symptoms and faults (Miguelanez, et al 2008). For example, a 'DoorAirFlowDeviation' symptom is associated with some 'DoorAirflowSignal' observation and some 'DoorAirflowFault', as shown in Figure 4.12.

DoorAirFlowDeviation:
core:refersToObservation ∃ DoorAirflowSignal

DoorAirFlowFault:
core:causedBySymptom ∀ DoorAirFlowDeviation
core:causedBySymptom ∃ DoorAirFlowDeviation

Figure 4.12 Observation Set for 'DoorAirFlow' Status

In a typical deployment, the reasoner is passed a set of instances in what is referred to as a *snapshot* of the current conditions of the system. The result is derived from the

realisation feature of the reasoner, i.e. finding the most specific class that an instance belongs to. Figure 4.14 illustrates this mechanism through the creation of the *DoorPressureSignal_A* instance (DPA001) and a *DoorAirFlowSignal* (DAF001). The two observations enable *Symptom* (SYM001) to be inferred as belonging to both *DoorAirflowDeviation* and *CompressorPressureDrop*. This second symptom enables the differentiation between the *DoorAirflowFault* (from Figure 4.12) and a *DoorActCompressorFault* (FAU001) as shown in Figure 4.13.

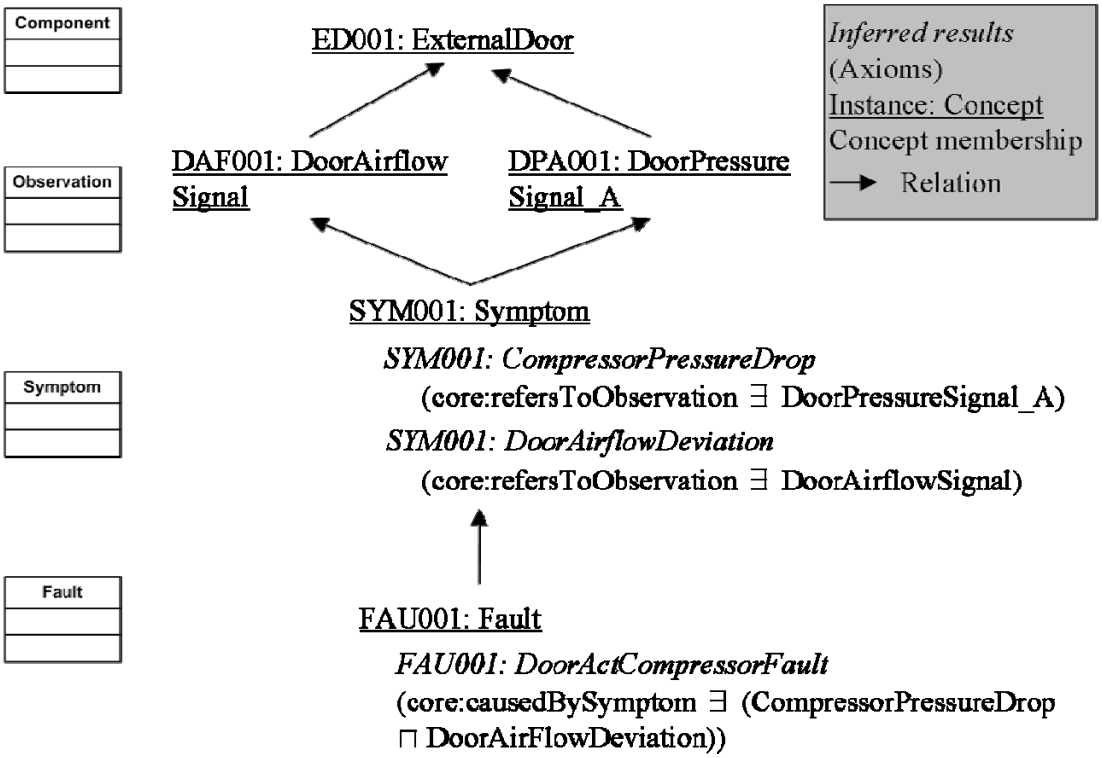


Figure 4.13 Generation of DoorActCompressorFault Inference Result

The limitation of this approach is that it does not cover the case where symptoms have temporal relationships. For example, it is feasible for the 'CompressorPressureDrop' symptom to be intermittent and therefore absent when the instances are passed to the reasoner, leading to the wrong conclusion. This type of information is considered to

be *tacit*, i.e. the domain expert knows of this phenomenon and can apply that knowledge during diagnostic activities.

The solution can be achieved in two ways, either the temporal characteristics can be queried during population and added to the properties of the instances, or the scope of the snapshot is extended to also cover historical observations employing a rule set to test for these relationships at run time. The second case is preferred here because it follows a knowledge driven approach using generic representation and interpretation methods, moving away from hard coded, application specific solutions.

Figure 4.14 represents the rule that finds the temporal relationship between a 'DoorAirFlowDeviation' symptom and a 'CompressorPressureDrop' symptom generating a 'DoorActCompressorFault'. This states that "if there is a symptom (?symp1) of type 'DoorAirflowSignal' and a symptom of type 'DoorPressureSignal_A' (?symp2) and ?symp1 occurs before ?symp2 then create a fault of type 'DoorActCompressorFault'".

```
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

[DoorFault1: (?a rdf:type ExternalDoor)
(?a hasObservation ?obs1)
(?obs1 hasSymptom ?symp1)
(?symp1 rdf:type DoorAirflowSignal)
(?obs1 timeStamp ?ts1)
(?symp1 refersToFault ?fault1)
(?a hasObservation ?obs2)
(?obs2 hasSymptom ?symp2)
(?symp2 rdf:type DoorPressureSignal_A)
(?obs2 timeStamp ?ts2)
lessThan(?ts2, ?ts1)
->
(?fault1 rdf:type DoorActCompressorFault)
]
```

Figure 4.14 Rule For Finding Temporal Relationship

The intention of these rules is that they are operated in conjunction with the OWL-DL reasoner in order to support a dual function. The DL reasoner infers the status of the system, whereas the rule reasoner applies context that is specific to the local operation or component/asset type, as shown in Figure 4.15. This leads to a generic solution where all subsystems are contacted in the same way, but the way that each is handled can be varied using the logic approach.

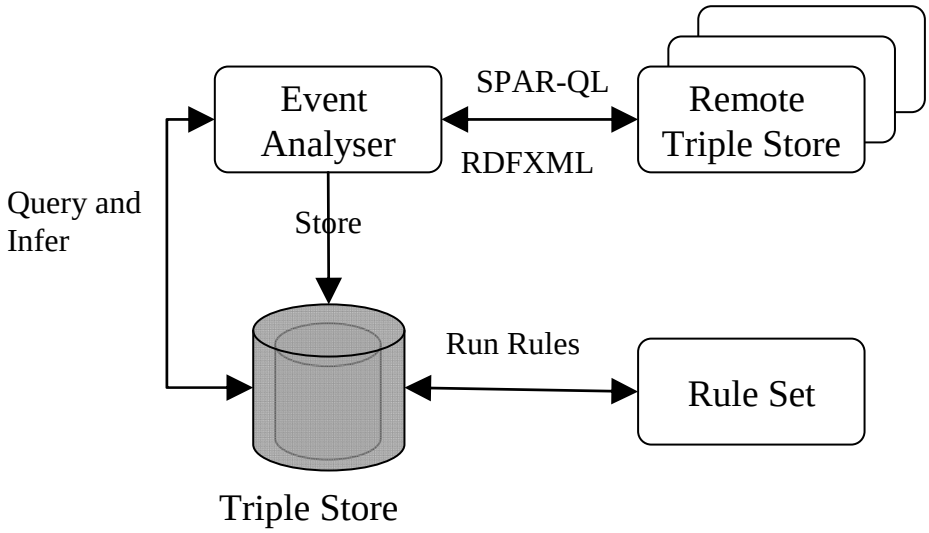


Figure 4.15 The Arrangement of Components for Rule Reasoning

4.6.1 Stakeholder Messaging

Event data from the legacy data systems is used to infer the criticality of any faults in the system. For the simplest case, a set of specialised classes is created to categorise the severity of faults. These are named 'CriticalFault', 'IncipientFault' and 'NonCriticalFault', as shown in Figure 4.16. Note that throughout this work a 'CriticalFault' is considered to be one that is *about to* cause a failure rather than one that *has* failed. These classes can be refined as required to suit the needs of the particular deployment within the system.

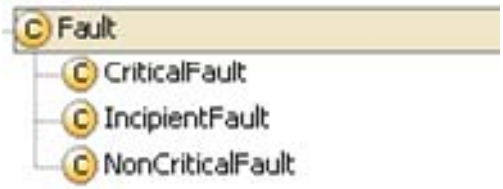


Figure 4.16 Set of Fault Categorisation Classes

These categorisation classes enable the creation of higher level categorisations. For example, a component instance with a 'CriticalFault' can be inferred to have a status belonging to the class of 'CriticalStatus' or specialisations of these classes. The non-monotonic nature of OWL ontologies supports this incremental approach to specialise concepts in a hierarchical view. This approach supports a second layer of reasoning that is removed from the detailed diagnostic rules. This is important in the multi-stakeholder, commercially competitive railway domain because it enables the exchange of information about the status of components without revealing how that status is attained.

The concepts at this level of the architecture form the content of messages exchanged. Figure 4.17 presents a class called 'SelfPoweredRailcarWithIncipientDoor' based on a *hasPart* property to 'ExternalDoorSystem', which has some 'IncipientStatus'. Higher level classes like this are intended for use by the stakeholder systems to form their own specialisations. For example, the train maintainer uses the relationship between the train location, temporal aspects of an observation, and the type of observation. Identifying some correlation between door failures and a particular stop (where the track characteristics cause the train structure to flex and interfere with the movement of the door) enables the train maintainer to ignore certain characteristics that might otherwise lead to further unnecessary investigation.

SelfPoweredRailcarWithIncipientDoor:

core:hasPart ∃ (core:ExternalDoorSystem ⊓ (core:hasStatus ∃ core:IncipientStatus))

Figure 4.17 Axiom representing 'SelfPoweredRailcarWithIncipientDoor'

4.7 Comments on the Ontology Design

The objective of the model resulting from this case study was to demonstrate that some features were made available to the stakeholders that are not available from a conventional systems design, such as one designed on a relational database. The semantic clarity of system concepts is an obvious feature of the ontology design. This is important in systems integration as the more ambiguity there is in the interface definition, the more time is spent fixing problems during deployment. There is significant supporting evidence of this since the problem has been the focus of research in many industries (Hunter and Liu, 2005).

Apart from semantic clarity, or lack of ambiguity, one of the promises of an ontology based approach is that it caters for the heterogeneity in data. The case study described by Damiani et al demonstrates the integration of two information systems and the need to integrate heterogeneous representation schemes (Damiani et al, 2005). The key concept described is that not every concept in one model will find a corresponding concept in the other. However, it is essential to have a mapping between these concepts for seamless integration of data from the two systems. The equivalent problem for the demonstration considered here is one where two similar trains have a slightly different physical structure resulting in different conceptual representation. Considering the case where a door control system in one vehicle

model features a single compressor for each door, whereas other train models either have a compressor supplying air to all doors in the vehicle, or no compressor at all because the doors are driven by electric power. In the case of the electric door it is a design choice to provide concepts to cover electrical components. One solution is to maintain the generic model as much as possible and only specify the electrical features in the observer/observation classes, leaving the remainder of the model fragments to represent more generic physical concepts.

For a door system made up of a door, door controller, a compressor controller and a door mechanism (which handles motion and limit switch signals etc), 'DoorSystemStatus' represents the status of the 'DoorSystem' and is therefore the focus of any queries of the condition of the door system. Figure 4.18 illustrates a fragment of the knowledge base modelling the concepts required to store these features.

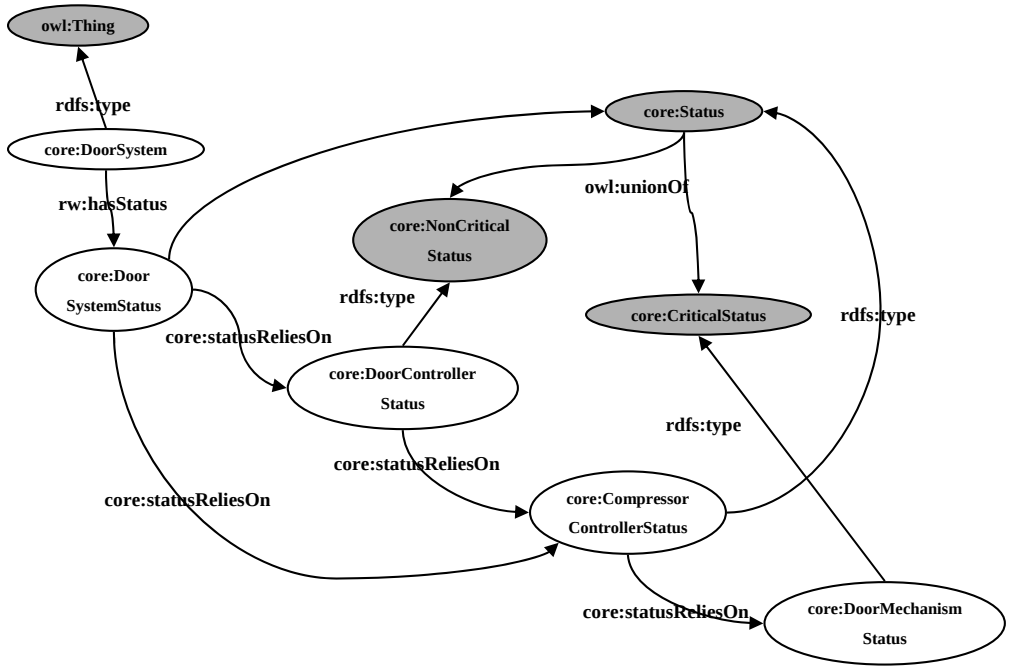


Figure 4.18 Fragment of Mapping of Instance Data to the Ontology Model

Considering the query class, “is there a door system with a door system status that relies on a component with a non-critical status, which in turn relies on a component with a critical status”.

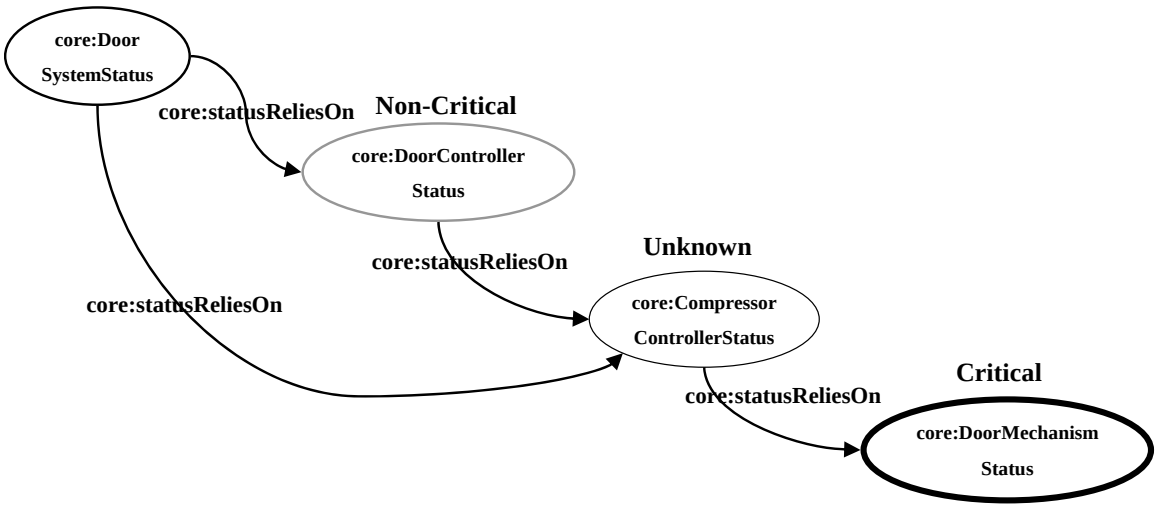


Figure 4.19 Demonstration of the Open World Assumption (OWA)

Figure 4.19 illustrates the case where there is a non critical ‘DoorControllerStatus’, the ‘CompressorControllerStatus’ is unknown, but the ‘DoorMechanismStatus’ is critical. Because the world is open in this repository, the lack of information about the ‘CompressorControllerStatus’ does not yield a negative conclusion. The model satisfies the logic of the query even though there is no information regarding the status of the compressor. In such a case it is considered that the DoorMechanismStatus captures the symptoms presented by a defective compressor controller being critical.

On the face of it, the example given does not present any features that differ greatly from an RDMS approach. It could be argued that adding a critical 'DoorMechanismStatus' to a data base would be sufficient to enable an event to be identified and acted upon. The point to note is that the model extract shown is part of a much larger model where context is being inferred from multiple properties. The ability to deal with heterogeneity in data is a concept that becomes very powerful in a system aimed at integrating data from various types of asset.

4.8 Conclusion

In the case study described here the design methods and data handling techniques supported by an ontology approach are demonstrated. The case of railway vehicle maintenance management is considered where there are multiple stakeholders that collaborate to deal with an event. The analysis is focussed on a scenario of medium complexity where the current case entails a dependence on a manual process of communication between stakeholders. The resulting use case describes a solution that implements an ontology design with associated architecture for reasoning functions. The resulting design supports the three functions for which an ontology based approach is applicable, i.e. to assist in communication between human agents, to achieve interoperability or to improve the process and/or quality of engineering software systems.

CHAPTER 5

RAILWAY INTERFACE DEPLOYMENT

5.1 Introduction

This chapter considers a real railway integration scenario for describing the relationship between the railway vehicle and track. It demonstrates how an ontology based design can serve to overcome the heterogeneity in data by forming common concepts for different applications to relate to. It is shown how the ontology can capture some of the tacit knowledge of the expert and tie that knowledge to data through logic statements. This chapter also identifies a limitation of the ontology approach in dealing non-monotonic logic concepts in that concrete data types can be stored but are not used during reasoning. Therefore a design decision is detailed where the ontology either contains a placeholder for the data value, which can be interpreted by separate rules such as semantic Web rule languages (SWRL), or an instance derived from the data values as they are stored in the knowledge base. For example, a measurement value may be compared to a predefined number and an instance of a concept is defined, such as *Too High*, *TooHot*, etc. This matter relates to subject of DL expressiveness and decidability introduced in Chapter 2. The former option maintains the theme of a generic approach to multi system integration where a rule mark-up language forms a concrete (XML based) rule syntax for the Web (Paschke and Boley, 2009). The latter option requires a bespoke coding which shares similarities with the manual intervention demanded by XML schema integration.

5.2 Railway Interface Scenario Investigation

The management of spatially and operationally related assets, as well as data about those assets, presents a challenge to large scale systems integration. In general, data about the physical arrangements of assets is stored independently of asset condition data, such as remote condition monitoring (RCM) data. Furthermore, data relating to various assets is not necessarily recorded or transferred according to any standard. The upshot of this is a dependency on the engineer to integrate, interpret and act on information while using tacit knowledge gained through experience to make the right decision for a set of circumstances. This presents a concern for the organisation as knowledge can often get lost in a dynamic business environment (Dzbor et al, 2000).

Previous integration efforts have proposed multi-database systems that implement *second order logic* to represent the dependencies between concepts (Lakshmanan et al, 1996). Borgida and Serafini describe the application of *description logics* to solve some problems of database integration (Borgida and Serafini, 2002). This work highlights the complexities in seemingly trivial integration. In this chapter an example of railway track measurement is considered to utilise the logic representation features in OWL-DL to represent the dependencies between physical components and associated data driven events.

Wheel Impact Load Measurement (WILM) systems describe a type of remote condition monitoring (RCM) system that detects the condition of the wheelsets and axles of railway vehicles from the trackside. These systems detect the forces of the individual wheel, the individual axle and the combined load of the vehicle. An important condition monitoring function is performed as a train with high wheel or

axle forces can degrade the quality of the railhead and supporting infrastructure and in the worst case can cause cracks in the rail and sleepers. Therefore, these systems identify vehicles with high wheel or axle forces, tagging them for inspection and maintenance. The integration of this data is usually limited to the attachment of vehicle identification data, either automatically through vehicle tagging, or manually through time/date and route data mapping from a scheduling database.

A Hot Axle Box Detector (HABD) system checks the temperature of the axles and wheels of railway vehicles as they pass. A detected hot axle box is an indicator of an impending bearing seizure which can lead to catastrophic derailment. A detected hot wheel is an indication that a brake shoe has seized. This could lead to damage to the track by the vehicle. Vehicles with components of increased temperature are tagged for inspection and maintenance whereas trains with vehicles deemed to be at a level that is considered a priority are removed from service. Legacy HABD systems are typically built on low level technology relying on modem telecommunications to transfer data to the decision maker. There is no automatic train identification so manual identification of the train is performed when an event is detected.

In the current UK railway system, making use of potentially valuable data from WILM and HABD systems relies on a number of manual tasks. These tasks rely on personnel to manually access the RCM systems to identify events and then access systems containing train movement and identification data to perform the integration. The result is a report that is manually transferred, either by fax or email, to the train maintainers to enable maintenance to be performed. The focus of the case study in the context of the wider European research is to replace the manual aspects of the

integration of event data and to remove any potential ambiguity in the integration. Equivalent research that has been undertaken outside of the UK highlights that the requirements for integration are Europe-wide. A previous case study aimed to integrate railway vehicle measurement data using 'networked checkpoints'. A prototype system incorporates a rule engine to validate received data and evaluate some output. The integration is based on some predefined system data-base model that enables the linking of measured data in logical and geometric order. This solution relies on a statically defined data model (Maly, 2005). In the semantic model approach, a reference model of the various asset and measurement systems is produced so that any measurements taken can be fused at the common measurement point. This existing work serves to demonstrate a broad requirement for a capability to monitor various types of train model, and this requirement will increase when demands to interoperate trains throughout Europe are met (ERRAC, 2002).

The combination of WILM and HABD data with other information enables the generation of a context of information. This context supports the querying of knowledge about the system status. For example, the Rolling Stock Operator (RSO) wants to assess which vehicles are most in need of wheelset maintenance to avoid incurring charges from the Infrastructure Operator (IO) for damaging the track. The Rolling Stock Maintainer (RSM) wants to allocate resources while planning and prioritizing maintenance tasks. The IO wants to make judgments of likely track condition of a route, or even a region, for life cycle costing (LCC) activities (Garcia-Marquez et al, 2008). The Infrastructure Maintainer (IM) wants to perform maintenance prioritization and resource allocation from an estimate of the amount of force (dose) that a section of track has been subjected to over a period of time.

Furthermore, an estimate of the effect of the vehicle condition on other measured assets is required. For example, to establish the effect on a point machine alignment and subsequent operation by the repeated passage of trains with wheel defects.

Prior to investigating the scenario for integration of data it was necessary to understand the scope of data required to deliver an integrated solution. There are four main data sources required for the demonstration, which include the WILM and HABD data, the train movement data and the train consist identification data. The train movement and train identification data are stored in legacy systems that are not easily interfaced. A thorough investigation of these systems enables an understanding of the requirements and challenges for real integration. The connection of these systems was not feasible due to the complexity of the interface[†]. Therefore, in order to maintain the focus on the integration of train condition data, the vehicle data was mimicked from a computer based simulation.

Each measurement system was linked to a physical location in the system, which is usually represented as a text name. The key information, which is enough to identify the basic "train with fault" condition, is the measurement, the asset (train) and the location point. However, to support the information requirements described above, more context based system information was necessary. For the purpose of a simple demonstration, a route was selected along which a number of vehicle measurement systems were located. Figure 5.1 represents a route between Cardiff in South Wales and Crewe in the North East of England.

[†] Concerns were expressed by the stakeholders as the data received would have been 'live' and therefore presented a risk since an event would have exposed the train operators' data to a party outside of the railway industry.

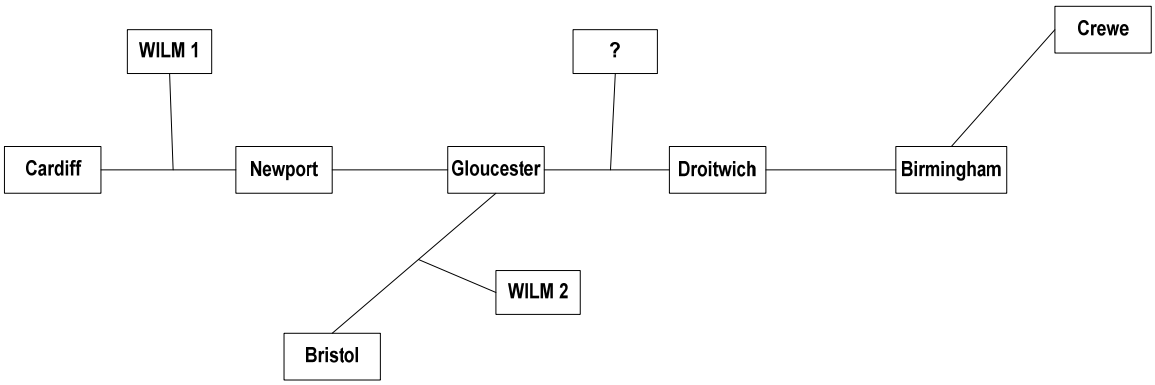


Figure 5.1 Railway Route with Measurement Systems

This route was selected because of the limited types of trains that use the route, enabling easier interpretation of the data. The location of monitoring systems that cover more than one route also made for a more interesting case study; the amount of force impacts that a track section without RCM received is interpretable from the measured trains and associated route information.

5.3 Analysis

5.3.1 Use Case Analysis

The use cases and use case models were developed from the roles of the main 'actors' as identified from the analysis activity. From a number of different use case diagrams and more specific use cases, a more general approach was made to constructing use cases applicable to not only wheel impact load detectors but also to any other rolling stock fault detector situated on the rail infrastructure, as opposed to on-board a train. Figure 5.2 describes the use cases that cater for the needs of the infrastructure maintainer to ensure that poorly maintained rolling stock assets are not damaging infrastructure assets. It also considers the need of the rolling stock maintainer to monitor vehicles and measure their impact on the track.

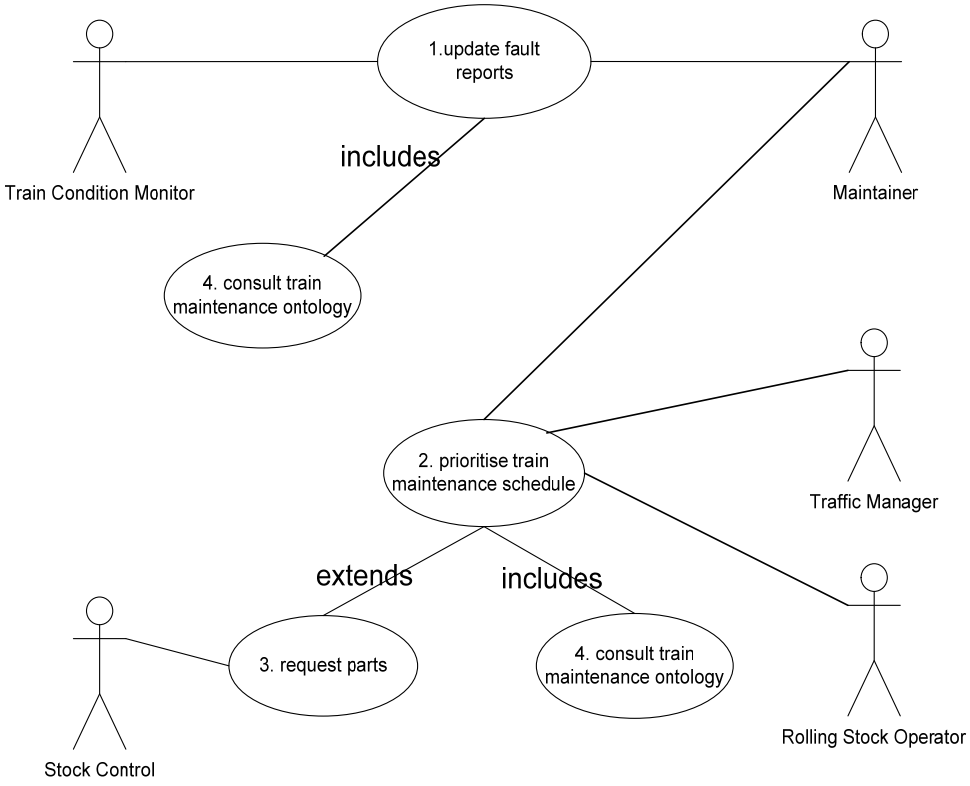


Figure 5.2 RS-Infra Use Case Diagram

For the purpose of demonstration the infrastructure is modelled such that any part of a route can be restricted to certain speeds. This is based on the interpretation of the level of damage that track has experienced through train movement. In support of this, a route is modelled as a sequence of route segments. Therefore, a route with some speed restriction of 50 mph is defined as either starting with a route segment, which is restricted to 50 mph (*RouteSegment50mph*) or containing some route segment, which is restricted to 50 mph.

Figure 5.3 illustrates the type of information that can be recorded in the information model. This information model is based on the operational viewpoint of the infrastructure and a requirement to retrieve information about the status of the track sections of the route and therefore the speed restrictions applicable to that route.

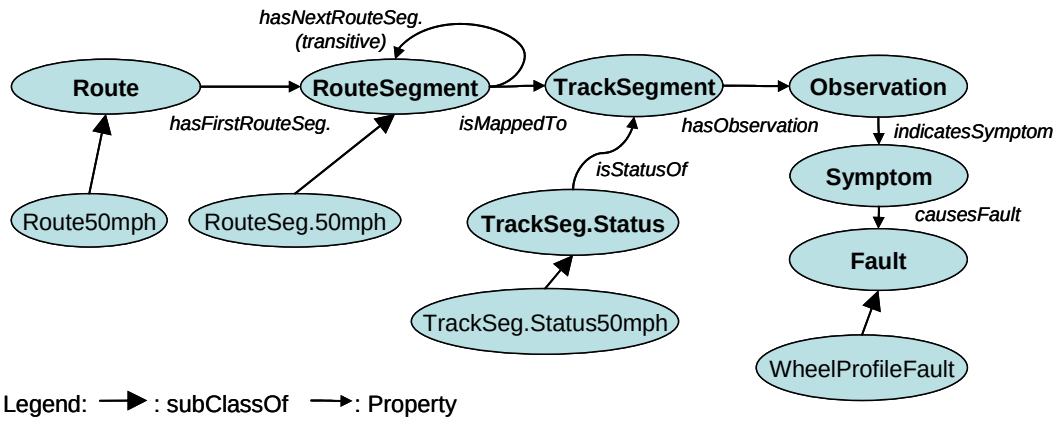


Figure 5.3 Modelling a route with speed restriction of 50 mph

A *RouteSegment* refers to routes in the real world that represent abstractions used for traffic management. *TrackSegments* represent real physical concepts of track that can be damaged, measured or maintained. Using these concepts as a foundation, *RouteSegment50mph* is defined with respect to the underlying physical track segment. This segment is restricted to 50 mph if the status of the corresponding track segment indicates a restriction to 50 mph. (*TrackSegmentStatus50mph*):

Route50mph ≡ *hasFirstRouteSegment some (RouteSegment50mph or (hasNextRouteSegment some RouteSegment50mph))*

RouteSegment50mph ≡ *isMappedTo some (TrackSegment and (hasTrackSegmentStatus some TrackSegmentStatus50mph))*

TrackSegmentStatus50mph ≡ *isStatusOf some (TrackSegment and (hasObservation some (WILMMeasurement and (indicatesSymptom some (WILMSymptom and (causesFault some WheelProfileFault))))))*

Figure 5.4 Modelling a route with speed restriction of 50 mph (DL syntax)

In the description logic syntax presented in Figure 5.4 *TrackSegmentStatus50mph* is defined with respect to the monitoring data collected for this track segment. This

enables the automatic derivation of the status of a track segment given a set of monitoring data through the specification of *Observations*, *Symptoms*, and *Faults*. *Observations* represent plain measurements, while *Symptoms* denote measurements that deviate from the expected behaviour. *Faults* represent a further abstraction defined with respect to a set of *Symptoms,* i.e. what has been observed. Therefore, a *TrackSegmentStatus50mph* is defined (in a simplified manner) as the status belonging to a track segment that has been measured with some symptom that indicates a *WheelProfileFault*. A more specific representation of this wheel fault will decide the degree to which the track has been damaged and therefore dictate the speed to which the traffic should be restricted. Note that the reference to historical data discussed in Chapter 4 can be addressed here; additional rule reasoning could review historical measurements and use this to generate a more complex fault that would trigger an equivalent speed restriction.

5.4 Ontology

The foundation of the ontology model devised to support this scenario is the core ontology concept. The requirement to model concepts that bridge the wheel - rail interface results in much broader ontology model requirements. The first modelling concept addresses the train concept. On first inspection a train is a simple concept that all stakeholders share an understanding of. On closer inspection, and in the context of data capture, a train represents a complex set of requirements to the various stakeholders. To demonstrate the capture of information within a model, requirements for the model are defined and, through some refinement, model concepts are represented in a way that is appropriate for later retrieval of instance data. For example, “a train with a vehicle that is displaying some fault and that fault is of a high

nature, is a train with a high maintenance priority". To represent this kind of statement in a model it is first necessary to have some representation of train components and the relationship between them. Figure 5.5 represents a train model that is constructed from a 'consist' that consists of some 'Train Configuration'. This definition is necessary because a train can consist of more that one train, and some 'Vehicle', which has any number of connected 'Vehicles'.

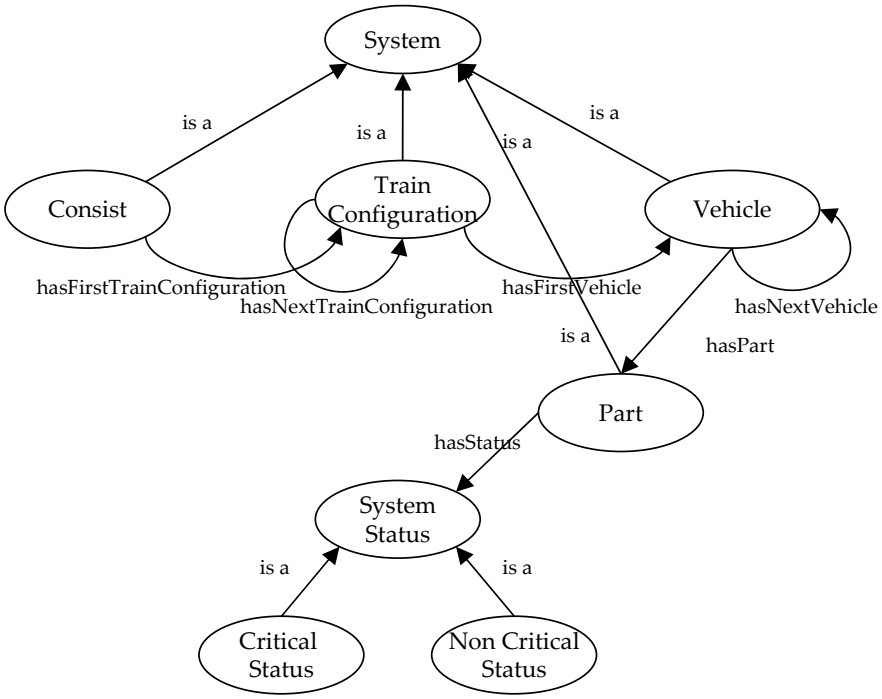


Figure 5.5 Simplified Model of a 'consist'

Since the exact composition of a train may not necessarily be known to the user, some method of querying the 'Vehicle' concept recursively is required. The 'transitive relation' in the semantic modelling approach supports this. To illustrate this concept consider three family members A, B and C who are related by a 'hasAncestor' property. If this predicate is transitive then A is related to C, even though this is not explicitly stated. Seidenburg and Rector have researched the application of the

transitive propagation technique in the medical domain (Seidenburg and Rector, 2006).

In the simplified model described in Figure 2, the 'hasNextTrainConfiguration', 'hasNextVehicle' and 'hasPart' are transitive relations. The specification of the model in this way enables an efficient method of identifying particular situations. For the classification of instances of "a train with a high maintenance priority", a classification axiom is created that allows the reasoner to 'realise' instances of this type. Therefore:

'A train with a critical status' {is equivalent to} =

hasFirstTrainConfiguration some

(hasNextTrainConfiguration some

(hasFirstVehicle some

((hasNextVehicle some

(hasPart some

(hasSystemStatus some

(CriticalStatus))))))) (1)

This type of axiom provides a convenient way of finding any 'TrainConfiguration' in a 'Consist' that has any 'Vehicle' that has some 'ComponentPart' with a 'CriticalStatus'. Therefore, when faults are categorised, by a similar axiom, those that do not affect the continued operation of the train are made equivalent to 'NonCriticalFault'. The benefit to the user is that the external query need only be addressed to 'a train with a critical status', so that the reasoner manages the discovery

of problem situations efficiently and equates them to this category. The work described here bears similarities to work of Fuchs et al where a model layer represents the context of the domain and the instance layer (or knowledge base) represents the state of the world with respect to this context (Fuchs, 2005). This means that knowledge abstracted from the source is expressed in terms of context information. In the work covered here the equivalent context meta-model is the RDFS structure for OWL concepts whereas the model layer is defined in the OWL DL language.

One of the benefits of the ontology approach is the extended querying that it provides. Typically, database systems based on schema designs define how data is structured. The limitation of this approach is that it is not possible to query the data structure once the database is deployed. In an ontology design, the classes and their properties are queried independently of, or in conjunction with, the instances. The railway model can therefore be queried to establish how classes are related to each other, even if no instance data has been generated. In fact, for some domains, instance data may never be added; this is because the interest is purely in defining the vocabulary and structure of the world, rather than the actual state that world is in. Deciding what is represented at the model level and what is represented at the instance level is an important part of the design process. In a domain such as the railway system, where there are multiple measurement systems producing large numbers of instances of data concepts, managing instance data is a very important consideration (Horrocks et al, 2004).

The measurements and other information that are collected represent the state of the world with respect to the context model. This is referred to as the knowledge base,

within which the knowledge is represented as statements about entities. These statements are formed by an instance of a property that forms a reference between one entity class and either another entity class or a data type class.

5.5 Terminology and Assertion Queries

The design patterns described in earlier sections of this thesis enable the creation of entity instances with reference to entity classes. These instances form a knowledge base, which can be queried in a number of ways. In RacerPro, querying the context model is referred to as T-Box (terminology box) querying which is the ontology model, whereas instance querying is referred to as A-Box querying (assertion box), (Harsleev and Möller, 2003). The ability to combine these query types is important for the decision support in a large and complex domain (De Giacomo, 1996).

Section 5.2 provides an overview of potential system interactions which will include the integration of various aspects of railway data. This overview is defined through an ontology concept that can store instances of asset data. This includes measurement data from various sources, including condition monitoring systems, such as a wheel impact measurement system. These systems are located at a point on the infrastructure and measure railcars as they pass. In the more complex case, the infrastructure maintainer wants to correlate vehicles detected with wheel force fault symptoms along a route to the status of the point machines that the vehicle has passed over. Answering this involves a multiple complex query pattern. To find this information, it is necessary to establish if there are any vehicles (T-Box) on a specific *Route* (A-Box) that have been measured with a *WheelFault* (T-Box). Then, an *InfrastructureElement* location (A-Box) needs to be identified for the *Observer* (A-Box) that made that *Observation* (A-Box). The location enables the identification of

the *TrackSegment* (A-Box) from the model, which identifies the related *RouteSegment* (A-Box), and then the *Route* (A-Box), as shown in Figure 5.6.

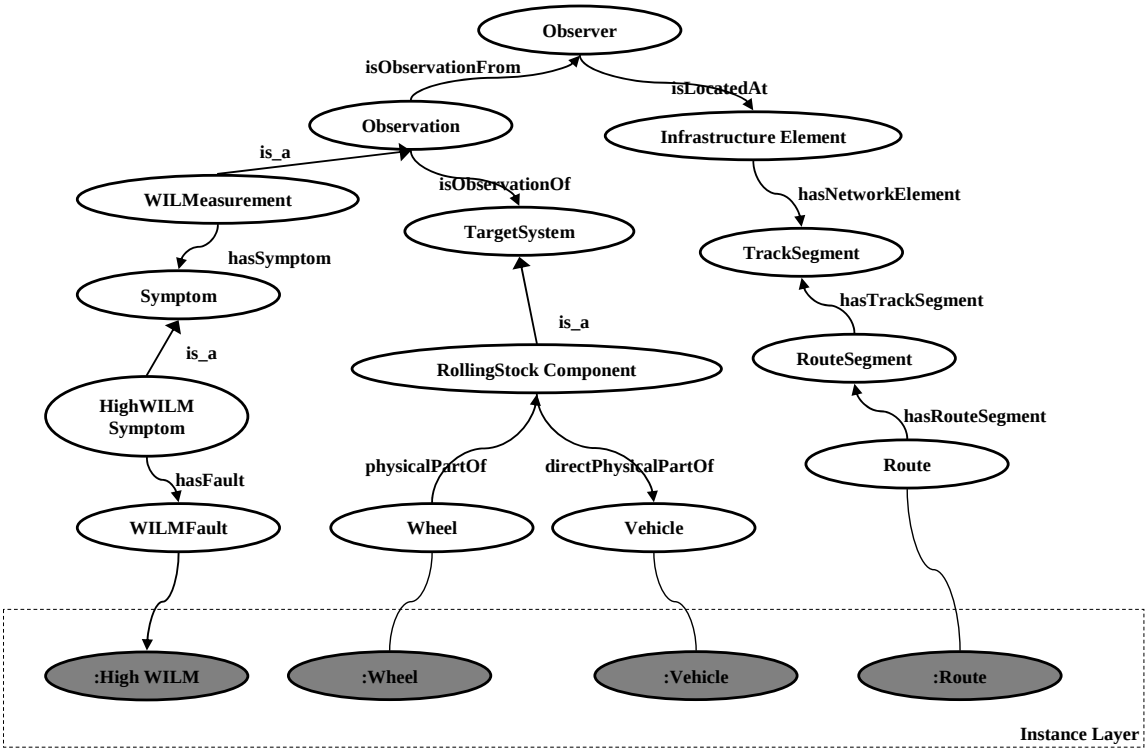


Figure 5.6 Schematic of the ontology querying process for vehicle - track interaction

Once the *Route* instance has been identified, the transitive features of the transportation network pattern are used to identify other assets along that *Route*. So each *PointMachine*, and its associated measurement, is identified by the instances of *Observation* classes and the *InfrastructureElement* class, by the sub-concept *PointMachine*, and the *TrackSegment* identified by the route segment, illustrated in Figure 5.7.

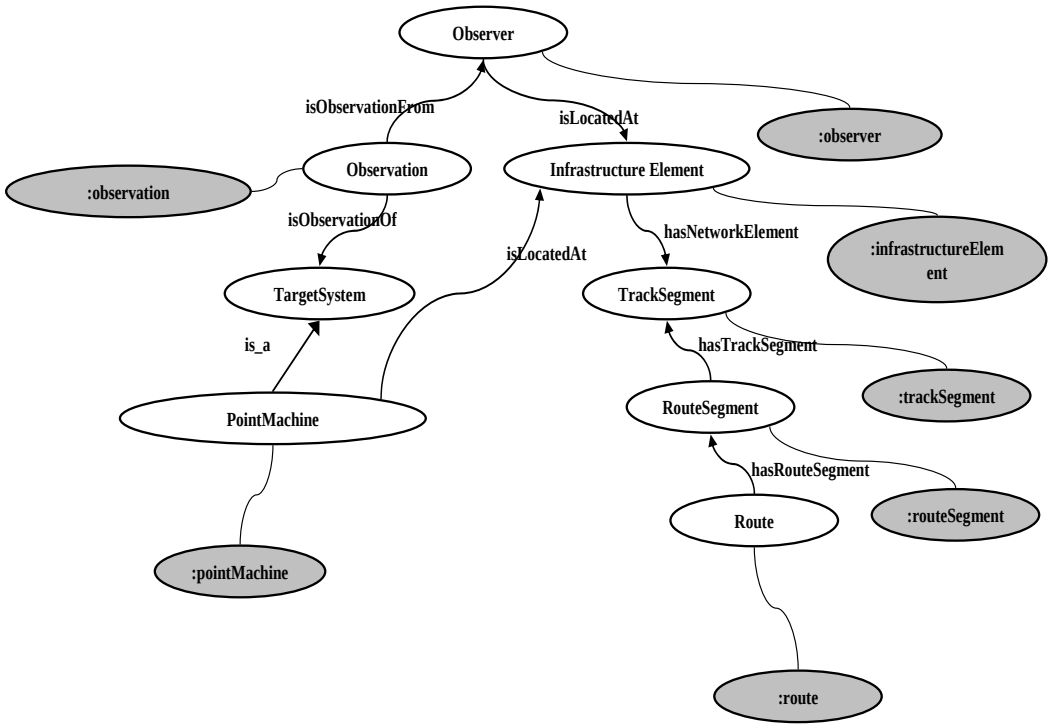


Figure 5.7 Schematic of Querying Process for point machine – track interaction

The information that is required to be transferred between applications is specified through SPAR-QL queries as described in Section 4.4.

The procedure for defining these queries is likely to follow a full use case analysis and is based on existing ontology resources. Figure 5.8 shows a query for retrieving RDF/XML in a selected format.

PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#;

```
PREFIX owl: http://www.w3.org/2002/07/owl#;
PREFIX core: http://www.integrail.info/ont/SP3A.owl#;

CONSTRUCT {
?train rdf:type core:PassengerTrainConsist.?train core:hasObservation ?o.?o rdf:type
?oType.
?train core:hasStatus ?status.?status rdf:type ?statusType.
?o core:hasSymptom ?s.?s rdf:type ?sType.
?s core:refersToFault ?f.?f rdf:type ?fType.}
WHERE {
?train rdf:type core:PassengerTrainConsist.?train core:hasObservation ?o.?o rdf:type
?oType.
?train core:hasStatus ?status.?status rdf:type ?statusType.
?o core:hasSymptom ?s.?s rdf:type ?sType.
?s core:refersToFault ?f.?f rdf:type ?fType.}
```

Figure 5.8 Sample query for the ontology interface

This example requests the result of any *PassengerTrainConsist* concept that has been attributed with a *Symptom* that is associated with a particular fault. This is a very general query and it is more likely that more specific queries will be required to find the components that are in a critical status. Figure 5.9 illustrates a more specific query that requests any trains that are of type *CriticalPassengerTrainConsist* and also the value of the observation from the WILM system that caused that *CriticalPassengerTrainConsist*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX core: <http://www.integrail.info/ont/SP3A.owl#>
PREFIX wilm: <http://www.integrail.info/ont/WILM_1.owl#>
CONSTRUCT {
        ?train rdf:type core:CriticalStandardPassengerTrainConsist .
        ?train core:hasObservation ?o . ?o rdf:type ?oType .
        ?o core:hasObservationData ?obsData . ?obsData rdf:type ?obsDataType .
        ?obsData wilm:hasWILMPeakLoad ?value .
        }
WHERE {
        ?train rdf:type core:CriticalStandardPassengerTrainConsist .
        ?train core:hasObservation ?o . ?o rdf:type ?oType .
        ?o core:hasObservationData ?obsData . ?obsData rdf:type ?obsDataType .
        ?obsData wilm:hasWILMPeakLoad ?value .
        }
```

Figure 5.9 Sample query for the ontology interface with values

In the prototype specification, it is desirable to incorporate the business logic within the knowledge base. The objective is to avoid the generation of domain specific code and provides an opportunity to support an interchangeable rule set associated with different asset types, i.e. each different train class is handled with a different rule set, or, each region has its own set of threshold rules.

Within the demonstration there are some clear requirements for business logic integration. For example, Network Rail's specification for fault handling "Control of Wheel Impact Forces" highlights that there are a number of different actions for handling faulty vehicles (Network Rail, June 2006). The application of the rules, set in documents of this type, form the business logic for rules within the knowledge base.

As discussed in Chapter 2, one of the limitations of a DL reasoner is the inability to handle datatypes such as integers and floats. OWL allows concepts to store values but the reasoner cannot reason over these values. OWL by itself can be used to represent information about units, but is not adequate for making the right inferences from the information. However, this is an area where SWRL can be used to great advantage (Elenius et al, 2009). Therefore, where any interpretation of specific values is required the data either needs to be parsed prior to entry into the repository or queried from the repository and processed by an application. Neither of these solutions follows the spirit of application independent data management. One alternative is create a placeholder for the data value in the ontology, which can be interpreted by separate rules such as SWRL (Horrocks, Patel-Schneider et al, 2004). The rule engine can periodically scan the instances for values that trigger a rule to fire (as referred to in

Chapter 4 sub-section 6). The rules can then generate instances of selected classes to be interpreted by the reasoner application during querying. This proposal requires an architecture with a combination of reasoning capability which has been successfully implemented for business rules in a commercial setting (Meech, 2010).

In the context of a prototype system, the application that delivers this rule function is referred to as the "Event Analyser". This is a reasoner based application that routinely answers SPARQL queries that will trigger reasoning to be performed. Figure 5.10 provides a simple example rule for handling wheel impacts of vehicles where the load value retrieved from the measurement system is compared to preset values. In each case a particular response level is defined as an A-Box concept.

```
[passConsFault1: (?a rdf:type core:CriticalStandardPassengerTrainConsist)
(?a core:hasObservation ?obs)
(?obs core:hasObservationData ?data)
(?data wilm:hasWILMPeakLoad ?load)
ge(?load 200)
le(?load 349)
->
(?a imain:hasResponseLevel imain:level1Response)
]

[passConsFault2: (?a rdf:type core:CriticalStandardPassengerTrainConsist)
(?a core:hasObservation ?obs)
(?obs core:hasObservationData ?data)
(?data wilm:hasWILMPeakLoad ?load)
ge(?load 350)
le(?load 399)
->
(?a imain:hasResponseLevel imain:level2Response)
]

[passConsFault3: (?a rdf:type core:CriticalStandardPassengerTrainConsist)
(?a core:hasObservation ?obs)
(?obs core:hasObservationData ?data)
(?data wilm:hasWILMPeakLoad ?load)
ge(?load 400)
le(?load 499)
->
(?a imain:hasResponseLevel imain:level3Response)
]
```

```
[passConsFault4: (?a rdf:type core:CriticalStandardPassengerTrainConsist)
(?a core:hasObservation ?obs)
(?obs core:hasObservationData ?data)
(?data wilm:hasWILMPeakLoad ?load)
ge(?load 500)
->
(?a imain:hasResponseLevel imain:level4Response)
]
```

Figure 5.10 Sample of fault train fault rules

5.6 Deployment

The selected case study implements a configuration that employs distributed reasoning as a key demonstration technology. The deployment of distributed reasoning relies on one node "knowing" that it can contact other nodes and retrieve reasoned results from those nodes. The solution to this requirement is provided by two methods. The first embeds the query within the node so that the node itself knows which query to send to the remote nodes. This requires the potential splitting of queries to separate them between differing nodes. The work of Zemanec introduces a number of approaches and describes a process that lets the user decide which node to send sub-queries to (Zemanek at al, 2008). The approach selected during this research relies on a third tier repository, referred to as a "central service". This node contains information about the types of nodes, their contact points and the queries that can be sent between them. In this case, the query does not need to be split as independent queries are provided for the various types of node being contacted. The approach demonstrates a more flexible arrangement as it enables queries associated with a node to be changed "on the fly" without re-deploying those nodes. This capability relies on rigorous testing of the query and the capability of the reasoner to infer correct results for those queries. This approach affords the benefit of reducing the requirement for application specific features, as the focus is on information rather

than the application. Figure 5.11 illustrates the arrangement of nodes that enable this approach and indicates each node's association with the central service repository. The event analyser application referred to previously sits at the top of this architecture.

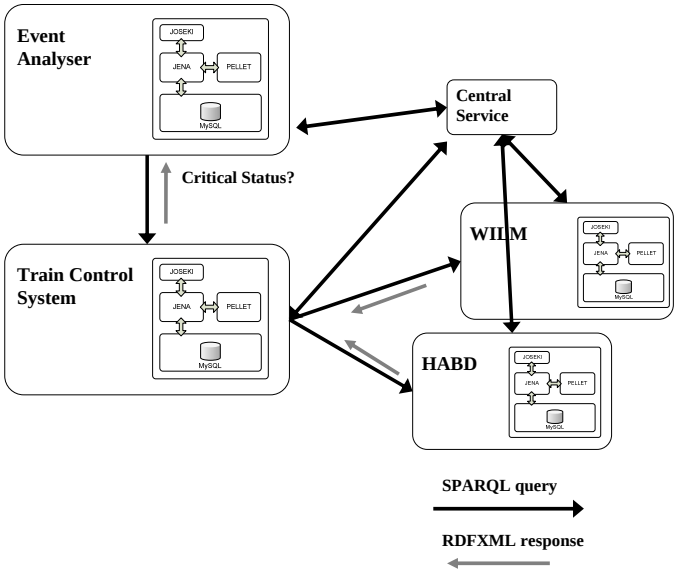


Figure 5.11 Candidate Architecture for Distributed Reasoning

The event analyser node will establish contact details of nodes available to it and will generate appropriate queries through contact with the query handler at the Central Service node. The Central Service is implemented and contacted through function calls to a Webservice called *CentralRepository*. Figure 5.12 illustrates the communication sequence between the components interacting to support the query and reasoning process. This diagram represents a UML sequence diagram with "swimlanes" indicating the level of communication. The detailed description of the purpose of the type of diagram is outside of the scope of this document; the purpose here is demonstrate the number of steps and reasoning layers required for the candidate architecture to function.

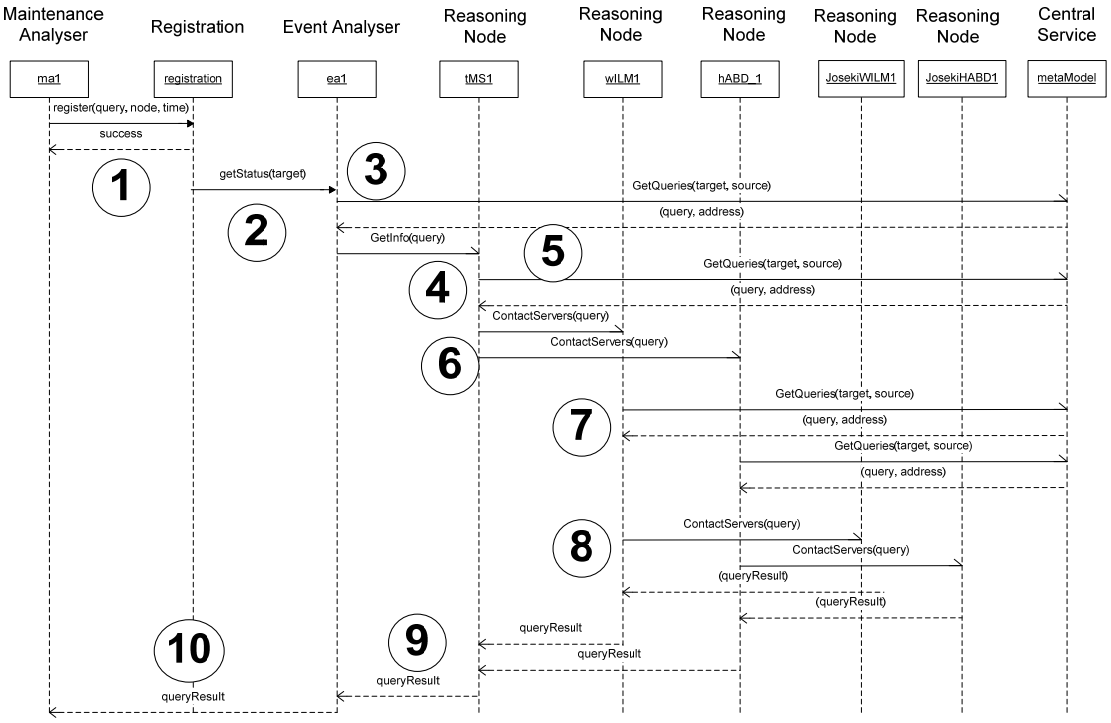


Figure 5.12 Sequence Diagram for Distributed reasoning

The sequence of interaction for the demonstration is as follows:

Step 1 – Query Registration

To initiate a query process the Maintenance Analyser or Operations Analyser registers a query with an Event Analyser node. This is driven by the application by selecting a specific query for an asset of interest and the time interval required for that query to be polled. The success or failure to register a query is dependent on the availability of the node of interest and the validity of the query for that node. This connection is made through the implementation of services that can discover each other using some Web Service Description Language (WSLD) discovery process.

This query process is driven by a simple method call to the registration process: Register(query, node, time). The registration class searches for appropriate Event Analyser resources to contact when the above method is called.

Step 2 – Get Asset Status

Based on the selected timing interval, the registration process contacts the event analyser application to get the status of an asset. This level is the most generic within the hierarchy and is based on (or some derivation of) the query…

```
"App", "criticalStatus", "PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX core:
<http://www.integrail.info/ont/SP3A.owl#> PREFIX id:
<http://www.integrail.info/ont/IMONObjectCondition.owl#> PREFIX
rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT ?o WHERE {?o
rdf:type id:CriticalVehicle }","PassengerTrainConsist_175006"
```

The logic behind this query states: return any vehicle that belongs to PassengerTrainConsist_175006 and is a CriticalVehicle. Note that a similar query can be applied for an Incipient Vehicle.

The ontology pattern used in this step is shown in Figure 5.13

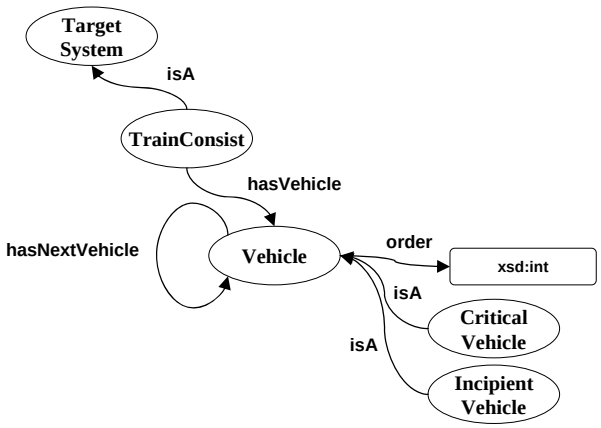


Figure 5.13 Train consist ontology design pattern

Step 3 Get Queries

Each event analyser will establish contact details of nodes available to it and appropriate queries through contact with the query handler at the Central Service node. This service is called Central Service and is implemented through the GetQueries method. The WSDL file for this service is shown in Figure 5.14. At the active point in the project this file was retrievable from the URL - http://147.188.146.68:8080/CentralRepository/CentralRepository?wsdl.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2-b05-RC1.
-->
- <!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2-b05-RC1.
-->
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.integrail.info/ont/GetQueries"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.integrail.info/ont/GetQueries" name="CentralService">
- <types>
- <xsd:schema>
<xsd:import namespace="http://www.integrail.info/ont/GetQueries"
schemaLocation="http://147.188.146.68:8080/CentralRepository/CentralRepository?xsd=1" />
</xsd:schema>
</types>
- <message name="GetQueries">
<part name="parameters" element="tns:GetQueries" />
</message>
- <message name="GetQueriesResponse">
<part name="parameters" element="tns:GetQueriesResponse" />
</message>
- <portType name="Central">
- <operation name="GetQueries">
<input message="tns:GetQueries" />
<output message="tns:GetQueriesResponse" />
</operation>
</portType>
- <binding name="CentralPortBinding" type="tns:Central">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="GetQueries">
<soap:operation soapAction="urn:GetQueries" />
- <input>
<soap:body use="literal" />
</input>
- <output>
<soap:body use="literal" />
</output>
</operation>
</binding>
- <service name="CentralService">
- <port name="CentralPort" binding="tns:CentralPortBinding">
<soap:address location="http://147.188.146.68:8080/CentralRepository/CentralRepository" />
</port>
```

```
</service>
</definitions>
```

Figure 5.14 Central Service WSDL File

This call is made by implementing the getQueries method in the following way:

```
Service svc = Service.create(svcQname);
svc.addPort(portQName,
SOAPBinding.SOAP11HTTP_BINDING,"http://147.188.146.68:8080/CentralRe
pository/CentralRepository");
 Dispatch<Source> dispatch = svc.createDispatch( portQName, Source.class,
Service.Mode.PAYLOAD);
```

Since the target resource for the query is not known prior to the result being

received from the Central Service, this call is configured to be made “on the fly”.

The ontology pattern used in this step is show in Figure 5.15.

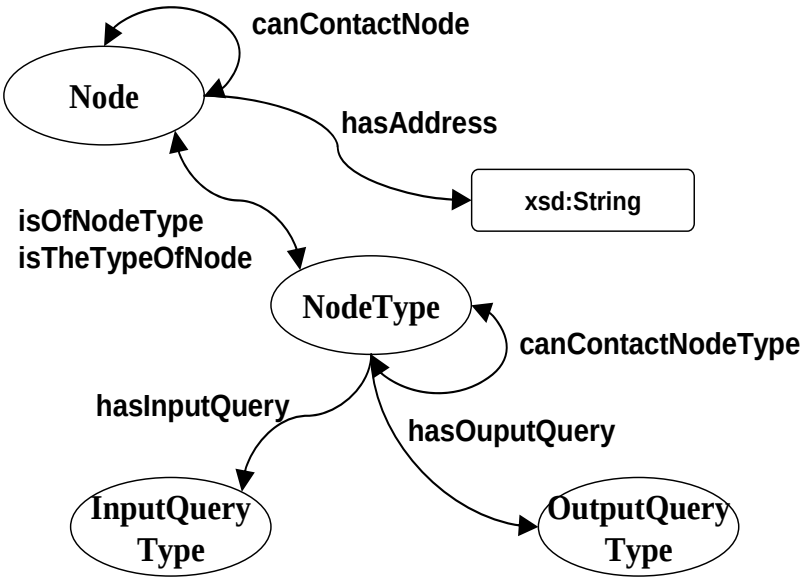


Figure 5.15 Node query pattern

Step 4 – Get Information

When the contact information and query is returned from the meta-model query handler, the EventAnalyser contacts the TMS node. It passes the query provided by the Central Service to this node through the Reasoning Node service.

In this case the query is:

```
"App", "criticalStatus", "PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX core:
<http://www.integrail.info/ont/SP3A.owl#> PREFIX id:
<http://www.integrail.info/ont/IMONObjectCondition.owl#> PREFIX
rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT ?o WHERE {?o
rdf:type id:CriticalVehicle }","PassengerTrainConsist_175006"
```

The WSDL file for this service is shown in Figure 5.16 and was available at location http://147.188.146.68:8080/ReasoningNode/ReasoningNode?wsdl.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!--
Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2-b05-RC1.
-->
- <!--
Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.2-b05-RC1.
-->
- <definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://www.integrail.info/ont/GetModel"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://www.integrail.info/ont/GetModel" name="NodeService">
- <types>
- <xsd:schema>
<xsd:import namespace="http://www.integrail.info/ont/GetModel"
schemaLocation="http://147.188.146.68:8080/ReasoningNode/ReasoningNode?xsd=1" />
</xsd:schema>
</types>
- <message name="GetInfo">
<part name="parameters" element="tns:GetInfo" />
</message>
- <message name="GetInfoResponse">
<part name="parameters" element="tns:GetInfoResponse" />
</message>
- <portType name="Node">
- <operation name="GetInfo">
<input message="tns:GetInfo" />
<output message="tns:GetInfoResponse" />
</operation>
</portType>
- <binding name="NodePortBinding" type="tns:Node">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
- <operation name="GetInfo">
<soap:operation soapAction="urn:GetInfo" />
- <input>
<soap:body use="literal" />
```

```
  </input>
- <output>
  <soap:body use="literal" />
  </output>
  </operation>
  </binding>
- <service name="NodeService">
- <port name="NodePort" binding="tns:NodePortBinding">
  <soap:address location="http://147.188.146.68:8080/ReasoningNode/ReasoningNode" />
  </port>
  </service>
  </definitions>
```

Figure 5.16 Reasoning Node WSDL File

The call is made through the implementation of the “GetInfo” method:

public String[] GetInfo(@WebParam(name="sourceType") String sourceType,@WebParam(name="abbrev") String abbrev,@WebParam(name="query") String query,@WebParam(name="params") String params)

Step 5 – Get Queries

The TMS node contacts the Central Service to establish the contact details and queries required for nodes available to it. This is a repeat of the mechanism described in step 3.

Step 6 – Contact Servers

The TMS node starts a thread for each of the satellite nodes, such as HABD and WILM with a query of the type defined in Figure 5.17.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX id:
<http://www.integrail.info/ont/IMONObjectCondition.owl#>
CONSTRUCT {
?vehicle rdf:type id:CriticalVehicle .
}
WHERE {
?vehicle id:isCriticalVehicle true .
}
```

Figure 5.17 Remote system query

Alternatively at this point, the implementation of open world features of the OWL language can be achieved. If the objective is to implement a more generic open world approach then the axiom defined in Figure 5.18 can be used to capture the dependencies between system components. This relies on the instantiation of the statusReliesOn dependencies between system concepts in the A-box. These dependencies can be optionally added for use as required.

```
<owl:Class rdf:about="#CriticalVehicle">
  <rdfs:subClassOf>
   <rdf:Description rdf:about="http://.. /Vehicle.owl#Vehicle">
  <owl:equivalentClass>
   <owl:Restriction>
    <owl:onProperty rdf:resource=" http://.. /Vehicle.owl#hasStatus"/>
    <owl:someValuesFrom>
     <owl:Restriction>
      <owl:someValuesFrom>
       <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
         <rdf:Description rdf:about=" http://.. /Vehicle.owl#NonCriticalStatus"/>
         <owl:Restriction>
          <owl:someValuesFrom rdf:resource=" http://.. /Vehicle.owl#CriticalStatus"/>
          <owl:onProperty rdf:resource=" http://.. /Vehicle.owl#statusReliesOn"/>
         </owl:Restriction>
        </owl:intersectionOf>
       </owl:Class>
      </owl:someValuesFrom>
      <owl:onProperty rdf:resource=" http://.. /Vehicle.owl#statusReliesOn"/>
     </owl:Restriction>
    </owl:someValuesFrom>
   </owl:Restriction>
  </owl:equivalentClass>
 </owl:Class>
```

Figure 5.18 Open World query class axiom

This query class implements the ontology pattern shown in Figure 5.19; for more detail on this implementation refer to Chapter 3 and specifically Section 3.5 - Explicit Versus Implicit Information.

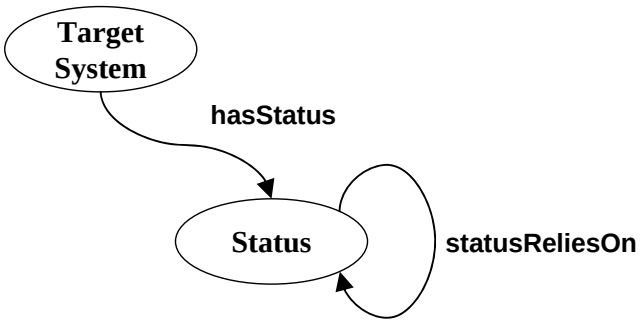


Figure 5.19 System Status pattern

Step 7 & 8 – Get Local Queries and Contact Servers

When the WILM and HABD nodes receive a query, they initiate querying of their own repositories. This is achieved via the Joseki service at each node. To enable this, the WILM and HABD nodes retrieve the appropriate queries from the Central Service as described in step 3. An example of the type of query that can be generated in the stage is provided in Figure 5.20.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX core: <http://www.integrail.info/ont/SP3A.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
CONSTRUCT {
?train rdf:type core:PassengerTrainConsist .
?train core:hasStatus ?tstatus . ?tstatus rdf:type ?tstatustype . ?tstatus core:isStatusOf ?train .
?train core:hasVehicle ?vehicle . ?vehicle rdf:type ?vehicletype .
?vehicle core:hasStatus ?vstatus . ?vstatus rdf:type ?vstatustype . ?vstatus core:isStatusOf ?vehicle .
?vehicle core:order ?vorder .
?vehicle core:hasPart ?wpart . ?wpart rdf:type core:Wheel . ?wpart core:isPartOf ?vehicle .
?wpart core:hasStatus ?wstatus . ?wstatus rdf:type ?wstatustype . ?wstatus core:isStatusOf ?wpart .
?item core:hasObservation ?itemobs . ?itemobs rdf:type ?itemobstype . ?itemobs core:refersToSystem ?item .
```

```
?itemobs core:hasValue ?itemvalue .
?itemobs core:hasSymptom ?itemsymptom . ?itemsymptom rdf:type ?itemsymptomtype .
?itemsymptomtype
core:refersToObservation ?itemobs .
?itemsymptom core:refersToFault ?itemfault . ?itemfault rdf:type ?itemfaulttype .
?itemfault core:causedBySymptom
?itemsymptom .
?itemsymptom core:hasEventLevel ?itemevent .
?itemobs core:isMostRecent ?itemAnnProp .
?lowEvent rdf:type core:Low .
?medEvent rdf:type core:Med .
?highEvent rdf:type core:High .
}
WHERE {
{ ?train rdf:type core:PassengerTrainConsist .
?train core:hasStatus ?tstatus . ?tstatus rdf:type ?tstatustype .
?train core:hasVehicle ?vehicle . ?vehicle rdf:type ?vehicletype .
?vehicle core:hasStatus ?vstatus . ?vstatus rdf:type ?vstatustype .
?vehicle core:order ?vorder .
?vehicle core:hasPart ?bpart . ?bpart rdf:type core:Bogie .
?bpart core:hasPart ?apart . ?apart rdf:type core:Axle .
?apart core:hasPart ?wpart . ?wpart rdf:type core:Wheel .
?wpart core:hasStatus ?wstatus . ?wstatus rdf:type ?wstatustype . }
UNION
{ ?item core:hasObservation ?itemobs . ?itemobs rdf:type ?itemobstype .
?itemobs core:hasSymptom ?itemsymptom . ?itemsymptom rdf:type ?itemsymptomtype .
?itemsymptom core:refersToFault ?itemfault . ?itemfault rdf:type ?itemfaulttype .
?itemobs core:isMostRecent ?itemAnnProp .
FILTER (?itemAnnProp = 'true'^^xsd:boolean) .
?itemobs core:hasValue ?itemvalue .
?itemsymptom core:hasEventLevel ?itemevent . }
UNION
{ ?lowEvent rdf:type core:Low .
?medEvent rdf:type core:Med .
?highEvent rdf:type core:High . } .
}
```

Figure 5.20 Sample WILM Query

In the case of HABD, the typical query is:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX core: <http://www.integrail.info/ont/SP3A.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
CONSTRUCT {
?train rdf:type core:PassengerTrainConsist .
?train core:hasStatus ?tstatus . ?tstatus rdf:type ?tstatustype . ?tstatus core:isStatusOf
?train .
?train core:hasVehicle ?vehicle . ?vehicle rdf:type ?vehicletype .
?vehicle core:hasStatus ?vstatus . ?vstatus rdf:type ?vstatustype . ?vstatus
core:isStatusOf ?vehicle .
?vehicle core:order ?vorder .
?vehicle core:hasPart ?apart . ?apart rdf:type core:Axle . ?apart core:isPartOf ?vehicle .
?apart core:hasStatus ?astatus . ?astatus rdf:type ?astatustype . ?astatus
core:isStatusOf ?apart .
?item core:hasObservation ?itemobs . ?itemobs rdf:type ?itemobstype . ?itemobs
core:refersToSystem ?item .
?itemobs core:hasValue ?itemvalue .
?itemobs core:hasSymptom ?itemsymptom . ?itemsymptom rdf:type ?itemsymptomtype .
?itemsymptomtype
core:refersToObservation ?itemobs .
?itemsymptom core:refersToFault ?itemfault . ?itemfault rdf:type ?itemfaulttype .
?itemfault core:causedBySymptom
?itemsymptom .
?itemsymptom core:hasEventLevel ?itemevent .
?itemobs core:isMostRecent ?itemAnnProp .
?lowEvent rdf:type core:Low .
?medEvent rdf:type core:Med .
?highEvent rdf:type core:High .
}
WHERE {
{ ?train rdf:type core:PassengerTrainConsist .
?train core:hasStatus ?tstatus . ?tstatus rdf:type ?tstatustype .
?train core:hasVehicle ?vehicle . ?vehicle rdf:type ?vehicletype .
?vehicle core:hasStatus ?vstatus . ?vstatus rdf:type ?vstatustype .
?vehicle core:order ?vorder .
?vehicle core:hasPart ?bpart . ?bpart rdf:type core:Bogie .
?bpart core:hasPart ?apart . ?apart rdf:type core:Axle .
?apart core:hasStatus ?astatus . ?astatus rdf:type ?astatustype . }
UNION
```

```
{ ?item core:hasObservation ?itemobs . ?itemobs rdf:type ?itemobstype .
?itemobs core:hasSymptom ?itemsymptom . ?itemsymptom rdf:type ?itemsymptomtype .
?itemsymptom core:refersToFault ?itemfault . ?itemfault rdf:type ?itemfaulttype .
?itemobs core:isMostRecent ?itemAnnProp .
FILTER (?itemAnnProp = 'true'^^xsd:boolean) .
?itemobs core:hasValue ?itemvalue .
?itemsymptom core:hasEventLevel ?itemevent . }
UNION
{ ?lowEvent rdf:type core:Low .
?medEvent rdf:type core:Med .
?highEvent rdf:type core:High . } .
}
```

Figure 5.21 Sample HABD Query

The queries returned to WILM and HABD nodes are applied to the respective Joseki services. The results are appended to the ontology models at these nodes and the reasoned function is invoked. The inferred results are then returned to the TMS node.

Step 9 - Query Result

The TMS node receives the result from the threaded satellite node and infers the result of the combined information. This result is returned to the Event Analyser application for display at the graphical user interface GUI.

Step 10 - Query Result

The result is returned to the top level service that initiated the query. At each stage in the process, the result from the previous layer is returned to the layer above. At the maintenance/operations analyser level, the results are combined to provide the overall requirements for actions and the result is returned for display or further

analysis. This layer uses the maintenance pattern shown in Figure 5.22 to achieve this integration.

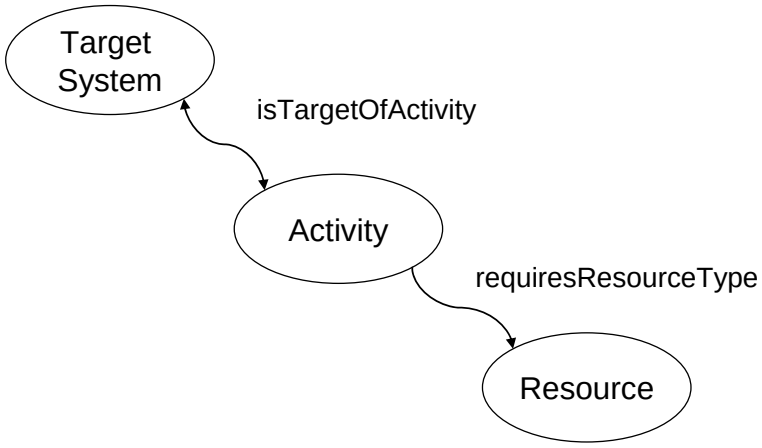


Figure 5.22 Maintenance ontology pattern

This demonstration of the deployment of a candidate architecture serves to demonstrate the relationship between the nodes and the ontology and reasoning features described in previous chapters. The key elements of this architecture are the Web services which enable a generic approach for each node. The features that differ between the nodes are the queries and the way that they are implemented, which differs from more conventional relational database oriented solutions.

5.7 Conclusion

The earlier sections of this thesis describe how the management of spatially and operationally related assets, as well as data about those assets, presents a challenge to large scale systems integration. This chapter presents further work aimed at addressing this challenge in the railway domain. A case study is described that

addresses data integration across the railway wheel-rail interface. A realistic architecture is described that supports the implementation of an ontology based system for integration of vehicle and track concepts. This chapter also identifies a limitation of the ontology approach in dealing non-monotonic logic concepts in that concrete data types can be stored but are used during reasoning. Therefore, where any interpretation of specific values is required the data either needs to be parsed prior to entry into the repository or queried from the repository and processed by an application. Neither of these solutions follows the spirit of application independent data management. The alternative described in this chapter creates a placeholder for the data value in the ontology, which can be interpreted by separate rules such as SWRL. The rule engine periodically scans the instances for values that trigger a rule to fire. The rules can then generate instances of selected classes to be interpreted by the reasoner application during querying. This enhancement to the system design allows concrete data values to be compared within the ontology structure without the need to create application specific code. The supporting architecture illustrates the process of querying between systems that reference a common asset within an ontology.

CHAPTER 6

MANAGING UNCERTAINTY

6.1 Introduction

The chapters presented in this thesis so far address the challenge of integration of data concepts from multiple heterogeneous data systems. The data concepts concerned represent known facts about the status of an asset or part of an asset. This means that retrieved information is integrated with other data to infer a concluding piece of information. In certain situations, the deteriorating status of an asset is not detected and a failure occurs. In such a case there is a requirement to use existing, possibly incomplete information, to manage uncertainty about the cause of a failure.

Semantic based reasoners do not currently cater for vagueness or uncertainty in information. This is an important requirement since in some circumstances it is not possible to have complete information regarding a particular event. Lukasiewicza and Stracciab describe work that aims at addressing this shortcoming through the extension of reasoning applications by a structured representation of non-monotonic, probabilistic interpretation (Lukasiewicza & Stracciab, 2008).

This chapter investigates the potential to apply probabilistic reasoning to the requirements of railway infrastructure monitoring.

6.2 Railway Context

Line side railway assets, such as point machines, form a fundamental part of the railway operation and have received considerable attention in the area of remote condition monitoring (RCM). For example, Roberts et al. describe a case study for fault diagnosis of assets at a railway junction (Roberts, Dassanayake, Lehrasab, & Goodman, 2002). Point machines provide the driving force for switching a set of points at a junction, which direct trains onto particular routes. Point machines are routinely appended with point heater systems, providing heat to the points mechanism in freezing conditions. These systems vary in technical complexity, ranging from those which provide simple on-off functionality to others that ensure the points are being heated through some basic closed loop function. In the latter case it is feasible to remotely monitor the operation of the heater.

In studying the economic justifications for investment, Marquez et al. highlight that RCM systems are not capable of capturing all of the failure modes (Garcia Marquez, Lewis, Tobias, & Roberts, 2008). The data collected during this study indicate that almost 15% of faults are logged as tested OK, i.e. the machine failed but by the time a maintenance team arrived on site it appeared to be functioning correctly. In such situations, the tacit knowledge of the maintainer is relied upon to assess the likely cause of the intermittent failure. Given the nature of the railway domain (the high costs of delaying trains) it is not uncommon for the maintainer to decide to not carry out any further investigation. In these circumstances the points machine remains in operation until either the fault re-occurs or some routine maintenance is performed.

Moving to a situation where available symptom and associated condition data is used to ascertain a failure is desirable. Increasing the certainty of a particular diagnosis leads to a greater likelihood of the corrective action being carried out. Demonstrating this approach requires the implementation of a number of applications where non-monotonic reasoning is applied to infer the most likely cause of a failure. The next section details the features of a prototype devised to demonstrate the approach taken.

6.3 Approach

An OWL ontology model is defined that represents system concepts and associated faults. The function of the model is to provide place holders for instances of observations made on assets. Figure 6.1 illustrates the proposed relationship between the 'Railway Asset' and subsequent observation and diagnostic concepts. Applications use the model to exchange data and interpret the effects of observations based on relationships between conceptual model components. This model is extended to cater for the situation where an instantaneous failure has been detected. The integration of information relevant to the failure is vital to this process. A meta-model of contactable resources provides the context of which information to consider for a particular case, i.e. that a point machine with a particular fault might be associated with weather and electrical data. Verstischel et al present a more detailed description of the requirements to separate this meta-level data from the system concept model (Verstichel et al., 2008).

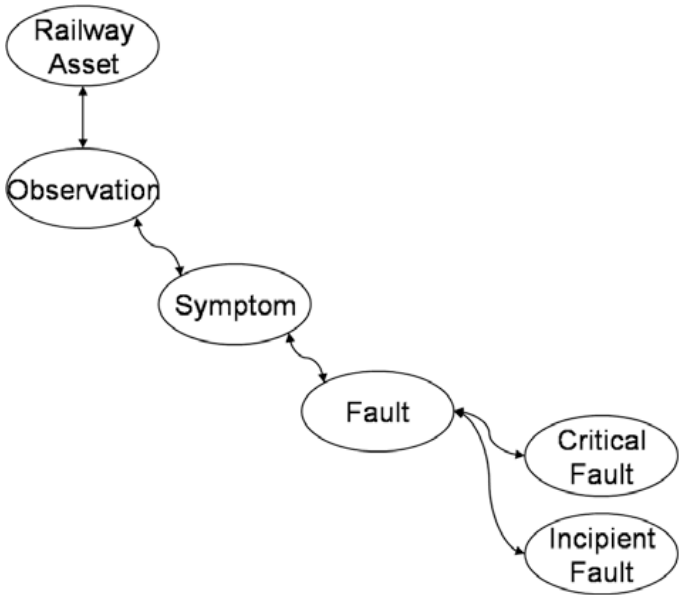


Figure 6.1 Fault diagnosis ontology meta model

For the case investigated, the ontology model provides the relevant concepts and the meta-model provides the resources to contact to retrieve the information. Fuchs et al. propose an architecture for semantic monitoring of large scale industrial systems (Fuchs et al., 2006). A similar approach is proposed in this work, where a network of web services use RDFXML to communicate the results of SPARQL queries based on railway domain data (Manola, Miller, & McBride, 2004; Prud'hommeaux & Seaborne, 2007). A 'semantic software stack' is used to handle data from legacy systems. Figure 6.2 illustrates how a MySQL database, configured to handle OWL data and referred to as a repository, is populated by a 'populator' application. Access to this repository is managed by the Jena API, which also handles interactions with the Pellet reasoner (Carroll et al., 2004, Sirin et al, 2007). A Joseki service, which provides a HTTP engine supporting the SPARQL protocol, is used to manage the querying of the repository (Joseki, 2003). The prototype architecture is based on a number of distributed 'nodes', each containing a software stack. Figure 6.3 illustrates

how the Joseki engine becomes the HTTP contact point for each node and enables querying of that node's repository.

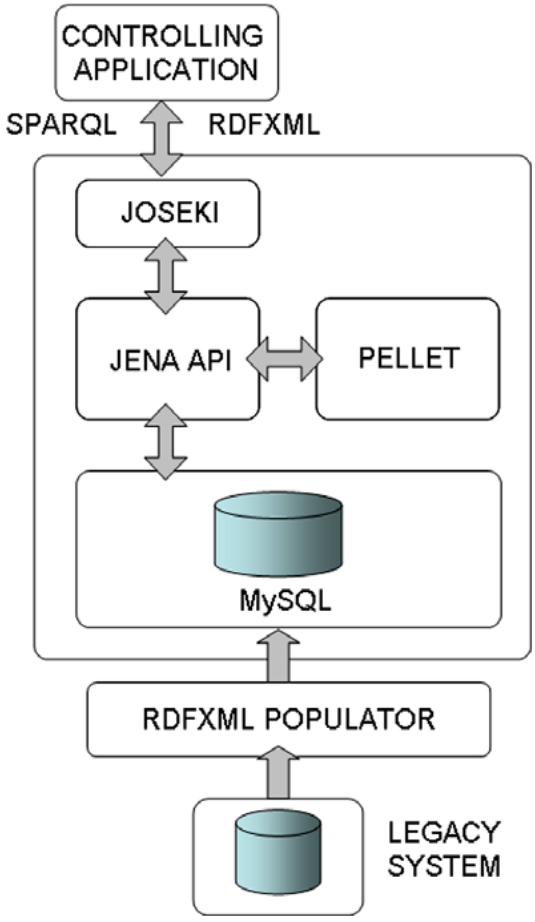


Figure 6.2 Semantic software stack

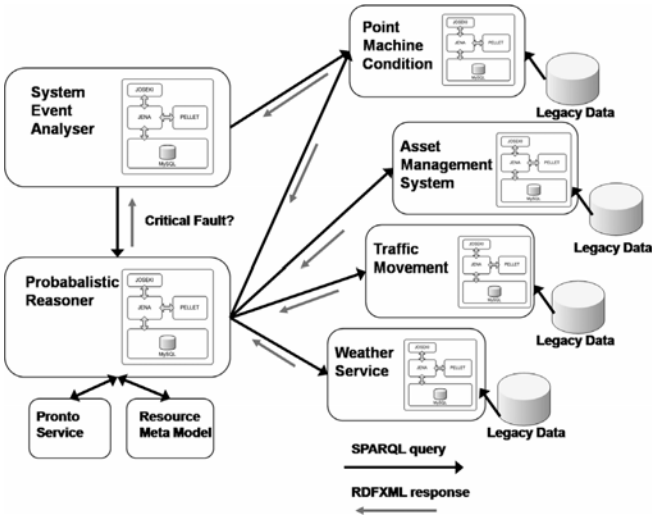


Figure 6.3 Prototype system architecture

At the legacy system layer, the reasoner is used to infer if a fault is 'Incipient' (i.e. faulty or defective) or 'Critical' (i.e. failed). In the case where a symptom has been diagnosed before failure occurs, an appropriate response is inferred by the querying system – the 'System Event Analyser'. In the case where the system has already failed, the 'Probabilistic Reasoner' application uses the meta-model of resources to establish which systems to query in order to entail the cause of the failure. In the current railway environment, domain experts are relied upon to decide the likely diagnosis for the failure and the appropriate course of action. The approach proposed represents a novel way of handling system failures where all available information is integrated in an effort to derive an appropriate diagnosis and solution. This novel approach requires an extension to OWL ontology reasoning that can cater for uncertainty in information. Probabilistic reasoners are intended to manage probabilistic uncertainty in OWL ontology models and therefore the architecture is extended with a probabilistic reasoning service. For the prototype system, the Pronto probabilistic reasoner was selected because it is based on OWL and has a convenient interface (Klinov, 2008). The results of the querying process are applied to the 'probabilistic reasoner' which infers the most likely response. The models required to support this application are described in the following section.

6.4 Models

Fault diagnosis is based on some model pattern representing the relationship between asset, asset status, measurement, symptom and fault. The arrangement enables the appropriate level of inference to be made based on the available data. Figure 6.1 illustrates how the fault concept is specialised as either 'Incipient' or 'Critical'. In the

case where the inferred fault is 'Incipient' the reasoner is invoked to infer further categorisations for preventative maintenance, i.e. the status is specialised to some characteristic concept.

The description logic axioms used in this reasoning are shown in Figure 6.4 where the symptom and event levels are provided by observations from the point machine condition legacy data. In the case where a fault is inferred as 'Critical', in this particular context it is proposed that it is too late to perform any preventative maintenance as a failure has already occurred.

Critical Fault = causedBySymptom $\exists$ (hasEventLevel $\exists$ High)
Incipient Fault = causedBySymptom $\exists$ ((hasTrend $\exists$ Increasing) $\sqcup$ (hasTrend $\exists$ Decreasing))

Figure 6.4 Description logic representation of fault status

In the real world, this case is considered to be an instantaneous failure where the user or controller of the system is aware of the situation due to inherent supervision of the system. This case presents the opportunity to use non-monotonic reasoning to support further investigation to assess the root cause of the failure. Equally this approach could be applied to test ''what if" analysis of the system behaviour. Probabilistic reasoning requires two models; the first represents a container for all of the information relevant to a particular point machine condition. The second model contains the generic probabilistic knowledge and concrete probabilistic knowledge that is required by Pronto to perform the probabilistic reasoning. El-Azhary, Edrees and Rafea propose a diagnostic method that defines domain knowledge as a domain ontology and domain models (El-Azhary, Edrees, & Rafea, 2002). In this work, the

domain ontology defines the language of the domain, whereas the domain models represent the particular viewpoint on the domain knowledge such that it is suitable for problem solving. The approach described here is similar; the ontology model contains the factors that enable the reasoner to infer the characteristics of the point machine whereas the probabilistic model supports the viewpoint of the domain knowledge. This approach supports the integration of information from remote resources as well as supporting interaction with the user to support 'what if' scenarios. Figure 6.5 shows the domain ontology that supports probabilistic reasoning. Note that the 'hasAssetFactor' axioms are used to infer this specialisation based on the response from the subsystems detailed in Figure 6.3.

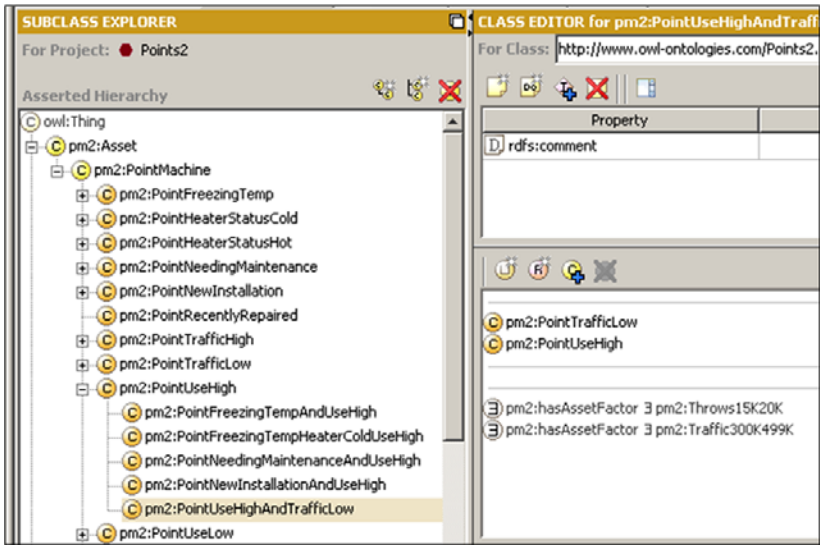


Figure 6.5 Ontology specialisation for 'PointUseHighAndTrafficLow'

In the example provided, these factors are between 300,000 and 499,000 trains and between 15,000 and 20,000 points throws. The non-monotonic nature of probabilistic reasoning provides a convenient mechanism for reasoning over incomplete and inconsistent information. Pronto enables the representation of two levels of expressivity, generic probabilistic knowledge and concrete probabilistic knowledge. For the generic case an expression of the type (D—C)[l, u] is used, where C and D are

description logic concepts and [l, u] is a closed subinterval of [0, 1]. This means that for a randomly chosen instance of C, the probability of being an instance of D is within [l, u]. For example, the probability of a point machine that is newly installed having some installation symptom is represented as:

(PointWithInstallationSymptom | PointNewInstallation) [0:9; 1:0]

i.e. the probability is between 90% and 100%.

In the case of concrete probabilistic knowledge the knowledge applies to a specific instance of data. Concrete probabilistic knowledge is represented in the form of a:X, where ‘‘a” is an instance and ‘‘X” is a general term restricted to the form (D— owl:Thing)[l, u]. For example, ‘PointMachine1’ is asserted with the fact that there is a high certainty that it was newly installed:

PointMachine1 : (PointNewInstallation | owl:Thing [1; 1]

i.e. PointMachine1 has a certainty factor that it is a ‘PointNewInstallation’
of 100%.

In the probabilistic model, the information is wrapped in OWL/ RDF tags enabling them to be interpreted by the Pronto programming interface – Figures 6.6 and 6.7.

```
<owl11:Axiom>
    <rdf:subject rdf:resource="#PointUseHigh"/>
    <rdf:predicate rdf:resource="&rdfs;subClassOf"/>
    <rdf:object rdf:resource="#PointWithInstallationSymptom"/>
    <pronto:certainty>0.0;0.05</pronto:certainty>
 </owl11:Axiom>
```

Figure 6.6 Example of generic probabilistic knowledge for point machine

```
<owl11:Axiom>
  <rdf:subject rdf:resource="#PointMachine_1"/>
  <rdf:predicate rdf:resource="&rdf;type"/>
  <rdf:object rdf:resource="#PointNewInstallation"/>
  <pronto:certainty>0.9;1</pronto:certainty>
</owl11:Axiom>
```

Figure 6.7 Example of concrete probabilistic knowledge for a point machine

6.5 Deployment

The models were deployed on two separate nodes, the first inferring the status of the point machine and the second inferring the likely cause of a failure. This prototype was used to test two cases: (i) In the case of incomplete information, where the point machine failed to operate, there is no data about the point heater status, but there is data about the local temperature implying that the fault has been caused by a ‘point with a frozen symptom’. It is demonstrated how the certainty of a diagnosis of ‘point with alignment symptom’ is increased over a freezing points failure through the entailment of the additional assertions – i.e. that the points have experienced heavy traffic and there has been a long period since maintenance, increasing the likelihood that a misalignment has occurred. (ii) In the case of inconsistent information, where a machine has recently been renewed, but the usage is high because the usage factor has not been reset in some external information system.

6.5.1 Incomplete information

It is feasible in certain circumstances for data to be unavailable to support a decision. As the number and complexity of information systems used by an organisation increases, so the likelihood of an incomplete dataset increases. This section provides an example of the type of scenario that can exist in a railway environment illustrating how deployment of a probabilistic application supports reason over uncertainty in data. The deployment of the application relies on a static probabilistic model and the dynamic distributed resources. The static model is represented as generic probabilistic knowledge in Figure 6.8. Where information is incomplete, assertions are made to cover the unknown conditions.

```
(PointWithFrozenSymptom|PointFreezingTemp) [0.55, 0.65]
(PointWithFrozenSymptom|PointUseHigh) [0.6, 0.6]
(PointWithFrozenSymptom|PointWithFreezingTempAndHeaterStatusCold)[0.9, 0.9]
(PointWithFrozenSymptom|PointWithFreezingTempAndUseHigh) [0.7, 0.7]
```

Figure 6.8 Generic expressions for point machine frozen symptoms

The dynamic concrete data is retrieved from the distributed resources in Figure 6.9. Information is retrieved from all nodes except the point heater monitor. The reason for this missing information may be attributed to a heater system that is unable to provide this information or is simply offline.

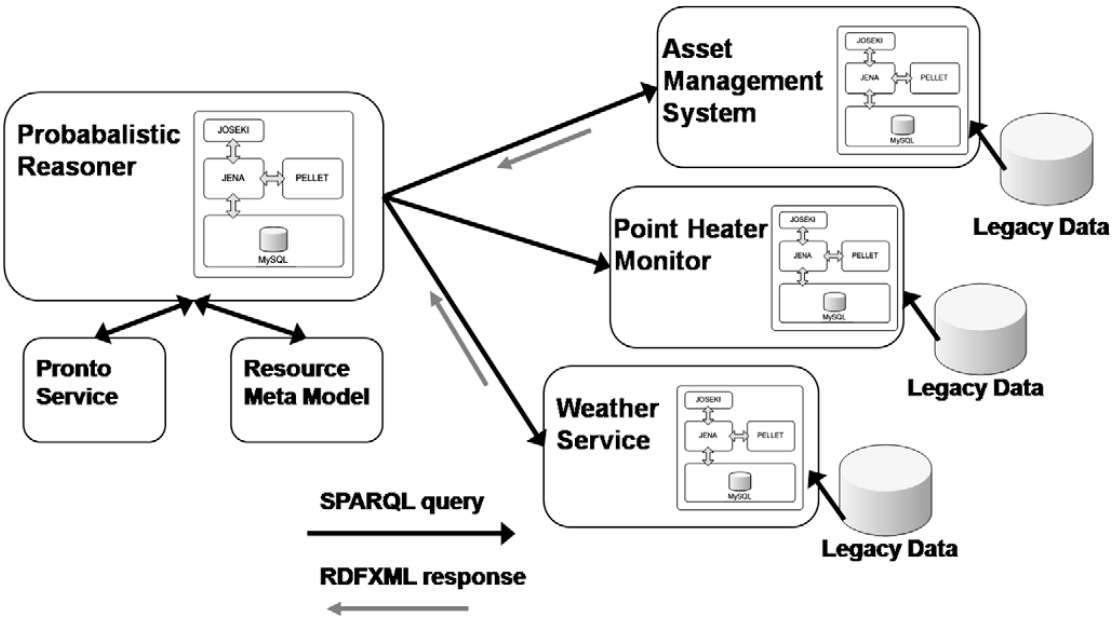


Figure 6.9 Resources queried in case of incomplete information

When the results are returned from the remote nodes they are added as certainty factors to the 'Probabilistic Reasoner' repository and the Pronto service is invoked by specifying the appropriate ontology model. The retrieved information suggests that the temperature in the vicinity of the machine is likely to be freezing, the point machine has experienced high use and the status of the heater cannot be ascertained. The concrete information that is provided to the reasoner is expressed as shown in Figure 6.10. This 'PointFreezingTemp' factor suggests that the source of the temperature data is not one hundred percent reliable or accurate.

```
PointMachine1: (PointFreezingTemp|owl:Thing) [0.9;0.9]
PointMachine1: (PointUseHigh|owl:Thing) [1;1]
```

Figure 6.10 Concrete probabilistic knowledge for the point machine

The fact that the heater status is not known is not asserted, but the lack of information is used to infer some appropriate outcome. From this concrete information the reasoner can entail the likelihood of the cause of the failure. The result from the process indicates that the fault can be attributed to 'PointWithFrozen Symptom' with certainty between 67.5% and 77.5%. Figure 6.11 illustrates the result achieved from the probabilistic reasoner.

Figure 6.11 Result of probabilistic reasoning for the point freezing symptom

This result is achieved by the reasoner overriding 'PointUse- High' and 'PointWithFreezingTemp' with 'PointFreezingTempAndUseHigh'. In reality, a domain expert may consider that, in light of the fact that the heater status is unknown, other factors may have caused the failure. For example, the point machine may be misaligned due to heavy traffic or maintenance maybe overdue. To support this assertion, a second set of generic properties are implemented as defined in Figure 6.12.

(PointWithAlignmentSymptom|PointNeedingMaintenance) [0.7, 0.7]
(PointWithAlignmentSymptom|PointTrafficHigh) [0.7, 0.8]
(PointWithAlignmentSymptom|PointNeedingMaintenanceAndTrafficHigh) [0.8, 0.8]

Figure 6.12 Generic expressions for point machine alignment symptoms

The certainty that a misalignment has occurred is assessed through the entailment of the concrete probabilistic knowledge asserting that the points have experienced heavy traffic and require maintenance. The concrete information that is provided to the reasoner is expressed as shown in Figure 6.13.

PointMachine1: (PointTrafficHigh|owl:Thing) [1;1]

PointMachine1: (PointNeedingMaintenance|owl:Thing) [1;1]

Figure 6.13 Concrete probabilistic knowledge for the point machine

Applying this concrete information to the reasoner indicates that the fault can be attributed to 'PointWithAlignmentSymptom' with a certainty of 80%. Figure 6.14 illustrates the result achieved from the probabilistic process. This case demonstrates how the features of Pronto are used to deal with the incomplete information and provide an appropriate answer. The result indicates that the 'PointTrafficHigh' and 'PointNeedingMaintenance' concepts have been overridden by the 'PointNeedingMaintenanceAndTrafficHigh' concept.

```
--------------------
Query : entail
Result: http://www.owl-ontologies.com/Points1.owl#P1:http://www.owl-ontologies.c
om/Points1.owl#PointWithAlignmentSymptom[0.8;0.8]
Explanation:
Explaining the generic constraint 36: (PointWithAlignmentSymptom|_TOP_)[0.8;0.8]
:
Lower bound is because of:
[[0: (PointWithAlignmentSymptom|PointNeedingMaintenanceAndTrafficHigh)[0.8;0.8],
 18: (PointTrafficHigh|_TOP_)[1.0;1.0], 16: (PointNeedingMaintenance|_TOP_)[1.0;
1.0]]]
Upper bound is because of:
[[0: (PointWithAlignmentSymptom|PointNeedingMaintenanceAndTrafficHigh)[0.8;0.8],
 18: (PointTrafficHigh|_TOP_)[1.0;1.0], 16: (PointNeedingMaintenance|_TOP_)[1.0;
1.0]], [18: (PointTrafficHigh|_TOP_)[1.0;1.0], 12: (PointWithAlignmentSymptom|Po
intTrafficHigh)[0.7;0.8]]]
```

Figure 6.14 Result of probabilistic reasoning for the point alignment symptom

6.5.2 Inconsistent information

In enterprise information systems, where there is multiple system interoperation and/or integration, there is a high likelihood of inconsistency in data. The dependency on temporal characteristics of data means that one system could be requested for data before another system has been updated. The example considered is that of a Maintenance Management System (MMS) where maintenance activities are updated in a database. The actual usage of the asset is stored in another system which means that some coordination and integration is required. If the data of the usage of an asset is not updated at the same time as the maintenance information, an inconsistency occurs. This section considers the case where an inconsistency between data indicating that a point machine has been newly installed, but also has high use, is considered. In the case of inconsistent information, the generic probabilistic knowledge is internally encoded in the ontology as shown in Figure 6.15.

(PointWithInstallationSymptom|PointUseHigh) [0, 0.05]
(PointWithInstallationSymptom|PointNewInstallation) [0.8, 0.9]
(PointWithInstallationSymptom|PointInstallationUseHigh) [0.75, 0.85]

Figure 6.15 Generic probabilistic assertions.

The querying system attempts to establish if the fault is based on a new installation symptom. This requires contact and querying of both the point machine condition service, to establish the usage, and asset management system, to establish if the point machine is a new installation, as shown in Figure 6.16.

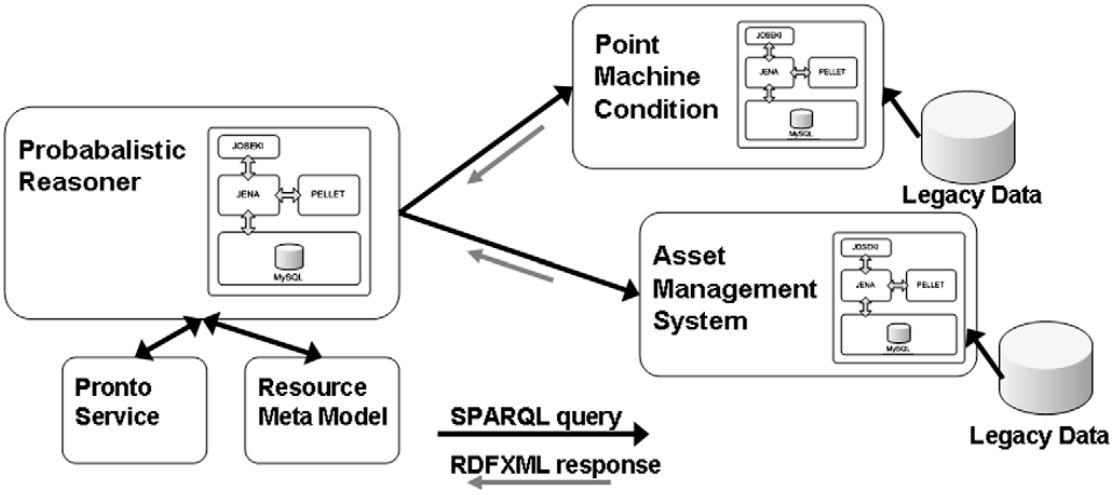


Figure 6.16 Resources queried in case of inconsistent information

Results returned from remote nodes are added programmatically as certainty factors and the Pronto service is invoked by specifying the ontology model to use. Figure 6.17 represents concrete probabilistic knowledge retrieved from the external resources.

PointMachine1: (PointNewInstallation|owl:Thing) [1;1]

PointMachine1: (PointUseHigh|owl:Thing) [1;1]

Figure 6.17 Concrete probabilistic knowledge for the point machine

```
--------------------
Query : entail
Result: http://www.owl-ontologies.com/Points1.owl#P1:http://www.owl-ontologies.c
om/Points1.owl#PointWithInstallationSymptom[0.8;0.85]
Explanation:
Explaining the generic constraint 38: (PointWithInstallationSymptom|_TOP_)[0.8;0
.85]:
Lower bound is because of:
[[17: (PointNewInstallation|_TOP_)[1.0;1.0], 4: (PointWithInstallationSymptom|Po
intNewInstallation)[0.8;0.9]]]
Upper bound is because of:
[[19: (PointUseHigh|_TOP_)[1.0;1.0], 3: (PointWithInstallationSymptom|PointNewIn
stallationUseHigh)[0.75;0.85], 17: (PointNewInstallation|_TOP_)[1.0;1.0]]]
```

Figure 6.18 Result of reasoning over inconsistent data

Figure 6.18 shows that the reasoner overrides the 'PointWithInstallationSymptomPointUseHigh', which produces a likelihood of a 'PointWithInstallationSymptom' of between 0% and 0.05%, with the 'PointNewInstallationUseHigh' concept. The result of this process indicates that the likelihood that a set of points is a 'PointWithNewInstallationSymptom' is between 75% and 85%. The aim of this process is to emulate the tacit knowledge of the domain expert, who might naturally assume that the points usage factor in the asset database had not been updated.

6.6 Conclusion

This work presents an approach for managing uncertainty in the event of an instantaneous failure of an asset in a large scale system such as the railway. It supports the emulation of the tacit knowledge of the domain expert to create assertions as to the likely cause of a failure. The proposed approach consists of two parts. The first is a network of distributed nodes that enable the querying of concrete facts about relevant factors in the domain. The second is a non-monotonic probabilistic reasoning process which uses the concrete facts to infer the most likely cause of the failure. A prototype system has been developed and two case studies have been defined to demonstrate its use. The first considers a case where there is insufficient information to come to a particular conclusion. The second considers how conflicting information is handled to produce an appropriate result. In both cases, it is shown how non-monotonic reasoning is applied to emulate the conclusion that a domain expert would provide. This works highlights how the approach could lead to an improvement in diagnosis in situations where available symptom and associated condition data is used. Increasing the certainty of a particular diagnosis leads to a

greater likelihood of the corrective action being carried out, which is valuable in a large scale system such as the railway. This work forms the foundation of more complex enterprise information system activities supporting the integration of data from multiple information systems.

CHAPTER 7

PROTOTYPE SYSTEM

7.1 Introduction

One of the original claims of this work was that, despite the array of enterprise level information management systems, large organisations still face a challenge in integrating data to support knowledge driven applications. The research aimed to address this challenge by providing a small scale demonstration of a real world application. The demonstration aims to illustrate the relationship between ontology modelling and a working application.

7.2 Working Demonstration

A prototype system was deployed as a series of user interfaces that allow the user to subscribe to a particular query and register that query against a data repository as shown in the sequence diagram in Figure 7.1. The result of the query allows the querying system to infer a conclusion from a result of the query. This simple function forms the foundation for a network of autonomous components that communicate using the technology described in this thesis. Fuchs et al, 2006 describe these components as layers of a functional architecture as shown in Figure 7.2. The query sequence diagram is embedded in the context management layer in this architecture,

i.e. the SPARQL queries referenced in Chapter 4 are deployed as query elements that search for context that the the reasoning service can use.

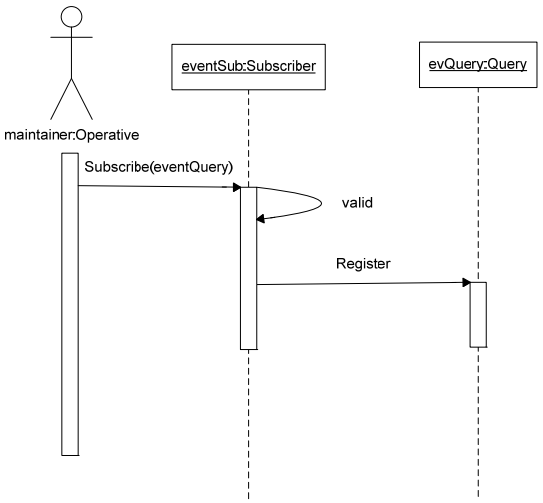


Figure 7.1 Sequence diagram for subscribing a query

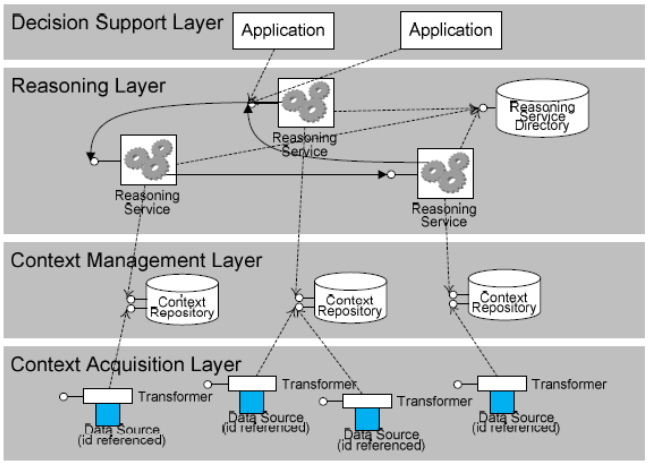


Figure 7.2 Functional layer of an ontology based system

A simplified version of this architecture was deployed as a prototype to demonstrate the capability of the technology. A Railway Domain Analyser graphical user interface (GUI) was designed to provide visualisation of the underlying concepts as shown in Figure 7.3. The interface consists of three functional screens. The first is the

resource management screen which provides the user with a view of the assets that are currently selected for review. This screen consists of a Resource Manager function that allows the user to configure queries and select resources to interrogate with those queries, a Rolling Stock Watch window that represents the concepts and designs described in Chapter 4, a Track Watch window that represents the concepts and designs defined in Chapter 5 and a Points Watch Window that represents the concepts and designs defined in Chapter 6. The second window, that is not shown, represents a Maintenance Analyser function that displays any maintenance activities that are inferred from events recorded by the measurement systems. The third window represents an Operations Analyser which effectively reviews the effect of any event on the wider operation of the railway network.

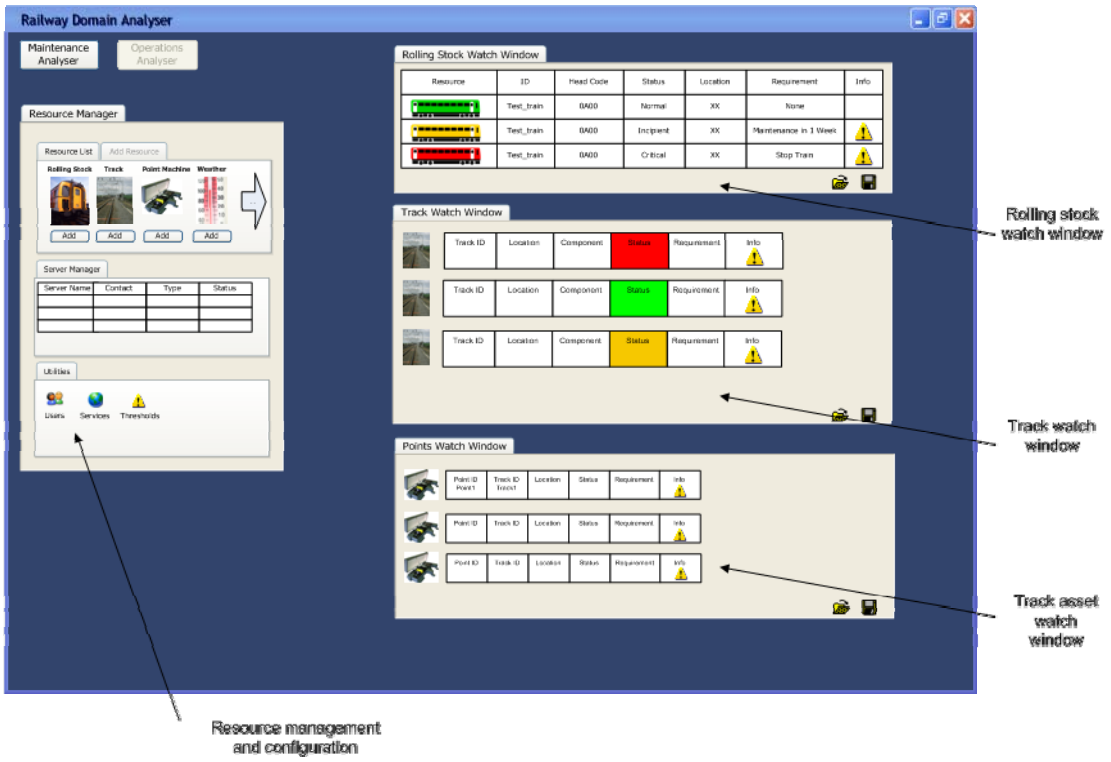


Figure 7.3 Railway Domain Analyser GUI

One of the challenges of capturing context in the railway domain is the process of collecting context information from experts. Through the research activities a number of anecdotal cases were established. One simple example was that of hot axle box detectors (HABD) that monitor the temperature of axle bearings and brakes of vehicles as the trains pass a measurement point. These simple systems consist of a data logger and temperature monitoring device that produce an alarm event on detection of an increased temperature. The reported problem with these devices is that certain train models have an exhaust manifold located in a position that can trigger an alarm on the logger. As a result the maintainer must carry out a manual step to remove those train types that trigger false calls to be removed from the maintenance plan.

A second simple case is one where wheel impacts from mis-shapen wheels are detected by a Wheel Impact Load Measurement system (WILM). The forces from wheels and axles on the entire train are measured at various locations throughout the network and decisions are made based on defined set of rules. This data can provide valuable information about the condition of wheelsets of trains when used effectively.The current usage enables infrastructure operators to monitor the condition of the vehicles and to stop or impede the speed of those trains with defective wheels [Network Rail, 2006]. Using the wheel and axle data in conjunction with other systems such as on on-board measurement data would enable a more holistic representation of the health of the vehicle. This information can then be used to create context for other systems.

The rules relating to the management of vehicles are detailed in railway standard GE/RT8000/TW5 which includes instructions on maintenance actions for wheel defects of different severity [Rail Standards and Safety Board, 2008]. These instructions formed an interesting case for elaborating the train maintenance ontology where ultimately the infrastructure operator is able to inform the train operator of a particular vehicle with a perceived fault.

A Vehicle Event Analyser (VEA) interface was therefore developed, shown in Figure 7.4, which incorporates a prototype data integration capability for railway vehicles. The VEA allows the user to visualise the status of a specific vehicle or a whole train that has been selected in the Railway Domain Analyser window. The reasoned results from data generated from vehicle mounted systems, axle temperature monitoring systems and wheel impact systems are displayed as a mimic diagram in the VEA window. The key function of the VEA is to allow the user to identify immediately any vehicles that require attention via a "traffic light" type system. The interface presents a mimic of each train currently being monitored and the status of each of the vehicle represented as *green* for 'OK', *amber* for 'requiring attention but still running' and *red* for 'requiring immediate attention and to be removed from service'. The VEA enables the user to oversee multiple vehicles by using the configuration function in the Railway Domain Analyser window.

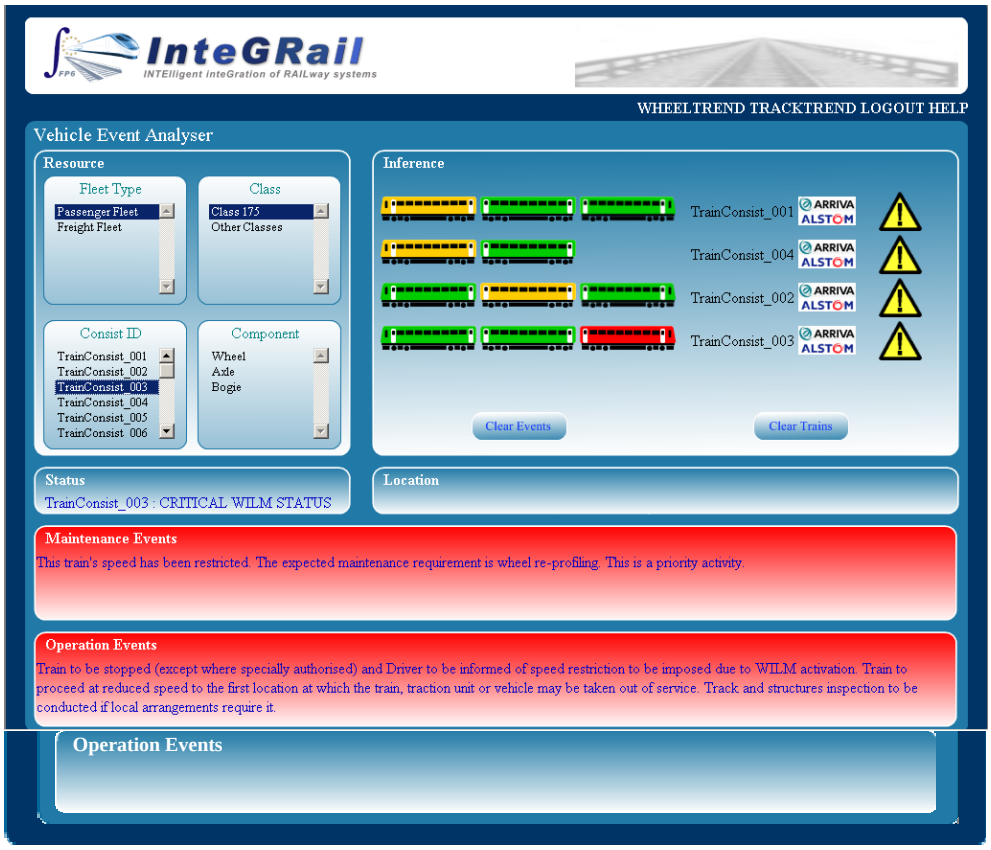


Figure 7.4 Vehicle Event Analyser GUI

The VEA window also contains Maintenance Events and Operation Events windows that display summary information relating to a particular vehicle that has been selected by the user. In the case where a vehicle is targeted for removal from service the related information is provided to inform the user of the intended maintenance actions and also any actions that will impinge on operations.

7.3 Decentralised Architecture and Evaluation

The VEA utilises distributed reasoning. The deployment of distributed reasoning relies on one node "knowing" that it can contact other nodes to retrieve reasoned results. The solution to this requirement was tested in two ways. The first was to embed the query within the node so that the node itself knows which query to send to the remote nodes. This requires the ability to split queries to send to different nodes. The second approach relies on a third tier repository, referred to as a "central service". This node contains information about the types of nodes, their contact points and the queries that can be sent to them. In this case the query does not need to be split as independent queries are generated for the various types of node being contacted. The second approach demonstrates a more flexible arrangement as it enables amendment of the queries associated with a node over time without re-deploying those nodes.

The functional elements that support the VEA function, depicted in Figure 7.5, require contact with the central service to maintain information on what repositories are available and which queries to use to interrogate those repositories. This arrangement shows where data is passed as SPARQL queries and RDFXML responses.

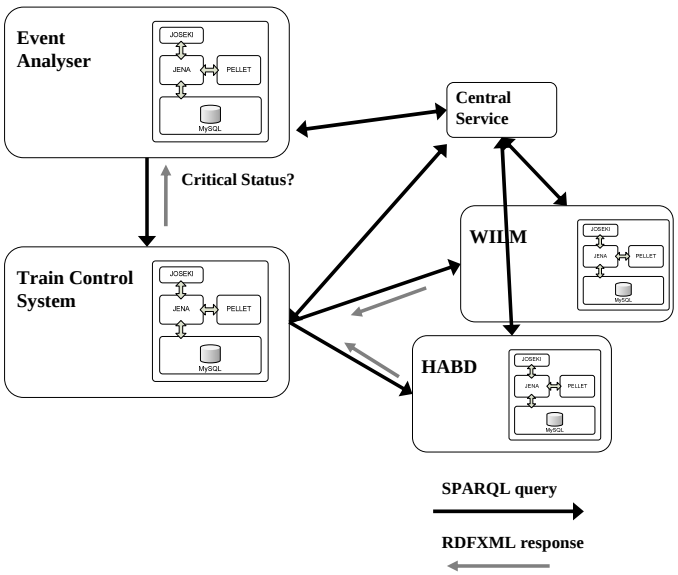


Figure 7.5 Decentralised architecture supporting VEA function

The complexity of the functions will vary considerably when deployed in the real world. However, the architecture developed forms the foundation for a system design that is extensible and based upon core ontology concepts. The common ontology features that support functions such as the VEA are depicted in Figure 7.6. These high level ontology concepts are general in that they do not relate to specific railway concepts. This allows the user to specialise the ontology design to suit the requirements of their specific application.

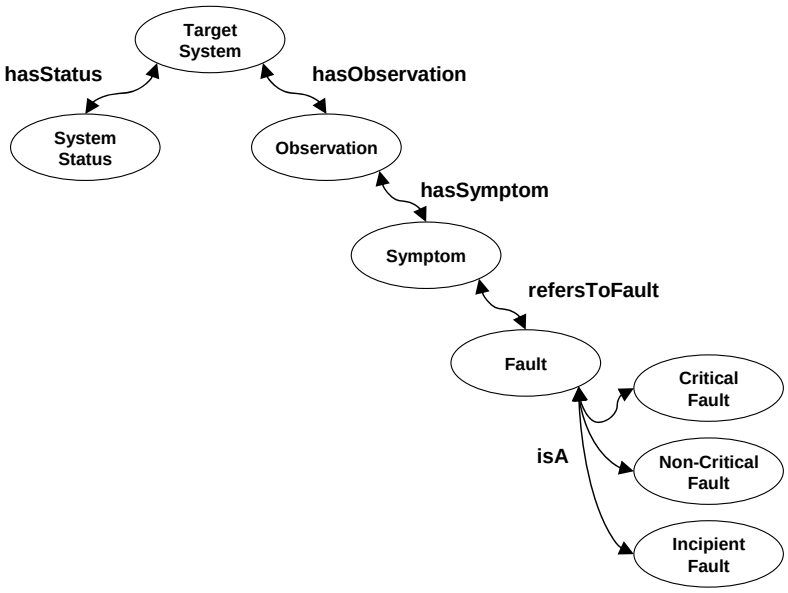


Figure 7.6 Core ontology concepts

The specific deployment of an ontology design allows the application designer to decide the level of detail to which they wish to interrogate the triple store. The manner in which these core ontology concepts are instantiated in the triple store reflects the manner in which resources are queried and the efficiency with which information is inferred. The efficiency of the reasoning functionality is a key factor in the performance of ontology based systems compared to conventional data storage and usage. The only way to test the efficiency is to deploy a decentralised network of reasoning nodes over a wide geographical area. This was feasible in the context of the overarching research project where numerous partners deployed nodes at various locations throughout Europe. Fuchs et al compare the performance of centralised and distributed query functions in the railway context [Fuchs et al, 2006]. These results conclude that a distributed network of query nodes is much faster as the reasoning load is balanced across the nodes.

7.4 Conclusion

This chapter describes the practical deployment of a prototype application layer that enables a potential stakeholder to understand the functionality that an ontology based system can provide. This application layer is based on a network of reasoning nodes that represent a distributed system and a graphical user interface for user interaction. The application serves to demonstrate how an alternative approach to conventional database applications might function and aims to show the benefits of a solution that is extensible. A review of the performance characteristics of the functions for querying and retrieving data is provided.

CHAPTER 8

CONCLUSIONS AND FURTHER WORK

8.1 Introduction

The motivation for the research presented in this thesis was the requirement of the railway industry to improve both IT system interoperability and decision making through improved availability of information derived from data. The aim of the work was therefore to demonstrate that integration is achievable by making heterogeneous data available to the decision maker through the application of the state of the art in Semantic Web technology. The needs of the stakeholders were constantly under review throughout the research so that the results would yield a practical solution to the end user.

This thesis provides a background to syntactic and semantic data exchange and interoperability leading the reader through an ontology design process for semantic modelling relevant to the railway domain. The early chapters of this thesis review the wide ranging concepts related to knowledge management for decision support and introduce a solution to information management that addresses the challenges of technical interoperability. The latter chapters of this thesis describe industry led case studies that promote a generic approach to data exchange and integration. The conclusions of the research are presented along with recommendations on areas that

require further work in order to allow technical interoperability to play its full role in improving railway system performance.

8.2 Conclusions

The European railway network is large, complex and multifaceted with each region's rail system operating at a different level of technical complexity. The physical interoperability of these currently independent systems presents challenges to every discipline from permanent way, signalling and power systems to rolling stock operation and maintenance. The systems integration that has occurred to date has been plagued by technical challenges.

The availability of the World Wide Web and numerous applications for data interoperability might imply that the requirements for data integration in the railway domain are easily met. However, research work referenced in this thesis has demonstrated that available web technology was not mature enough to support true interoperability. In particular two efforts to perform integration and interoperability have been highlighted in Chapter 1 – 1.1 Background. The first is the case of Engineers' Workbench undertaken by Atkins on behalf of Network Rail. The ambition in this project was to centralise all of Network Rails' Remote Condition Monitoring (RCM) data in one large repository such that all data was easily accessible for decision support activities. The difficulty reported at the time was that the XML Schema developed as the interface to numerous heterogeneous RCM systems had to be edited in collaboration with the data providers such that the semantics of the data were agreed through interaction. This process had to be repeated for each RCM system creating significant additional work effort. The resulting repository neither

matched the original data structure nor was not strong enough to enable an efficient query and retrieval process. The second case was formed by a consortium of railway operators with a common goal to integrate rolling stock data. This work aimed to demonstrate how the development of a coding standard for data exchanged by railway systems (both vehicles and installations), based on functional or physical breakdown, can greatly improve maintenance. Data collected by the system was to be fed to a diagnostic database, where it would be stored, organised and made available to logistic applications. The project, called Euromain, demonstrated that addressing interoperability issues by syntactic modelling had limitations. While a working system was successfully deployed, the conclusions were that centralisation of heterogeneous data between multiple stakeholders was not the answer as the interfaces were too rigid and the resulting data too difficult to interrogate once centralised (Shingler & Umiliacchi, 2003).

The conclusion to these investigations was that data centralisation was not the answer. The proposed solution in both cases was to maintain the data as distributed resources and use a common model of the data to efficiently retrieve the data required to perform a specific function.

The work detailed in this thesis demonstrates that OWL-DL ontologies can support the interoperability of information systems in the railway domain. It is shown through descriptions and examples how an ontology design can be constructed. The constituent parts of a design that support a Semantic approach are described in the context of railway scenarios. The unique features that the solution brings are highlighted in each case giving the reader an insight into the differences between an

ontology based approach and a solution based on a conventional database design. The work of Motik et al is presented where example models of the human body are discussed (Motik et al, 2006). Representing a complete decomposition may create an undecidable model which may require additional reasoning functions. This means that decisions need to be made upfront to consider the limit of details or resolution that the target model requires. This is akin to modelling the railway domain where the relationship between the level of model detail and the successful entailments presents a potential challenge to even relatively simple model designs. The activity of ontology model development requires a fundamental understanding of DL theory, which may preclude the contribution to model development by domain experts. Therefore further work is required to underpin the model design process and, specifically, ensuring that resulting models are decidable.

OWL-DL was used as a language to capture the formal semantics of the domain. Those semantics were used to infer knowledge from associated data recorded by measurement systems. The examples provided serve to demonstrate that OWL-DL is less ambiguous in describing data concepts than alternatives such as XML. Ontologies are therefore proposed as a candidate technology for supporting a semantic approach to railway data integration and decision support.

The major, novel contribution to the subject is the development of reasoning based applications that implement core and extended ontology design. These were purpose built for the railway domain and did not exist prior to the research. The intention was to demonstrate to potential stakeholders the purpose and benefits of ontology driven

information exchange. The focus being on inferring explicit information from implicit data. This is a difficult task when the stakeholders are not fully

The examples are provided as part of a network of reasoning nodes that support distributed data storage and reasoning and centralised knowledge based on ontology. The contribution in terms of outcome was the realisation by industry stakeholders that a Semantic approach was a realistic solution to the interoperability challenges that they faced. While the research is far from a complete solution, it makes some headway in explaining the processes and highlighting the technical requirements.

The specific outcome of the work are:

- A railway domain ontology – at the point in time when this this research was undertaken this outcome represented the first railway specific ontology resource aimed at information interchange.
- A railway rolling stock demonstrator applications that utilised a network of semantic nodes and exchanging instance data related to the railway domain ontology and application ontologies that extended from it. These applications used a reasoner function to infer railway decision support information.
- A journal paper that addressed the limitation of conventional description logic models by extending them with probabilistic reasoning functions to capture tacit domain information.

The key selling point of the research is that the data no longer needs to be centralised. The data can remain distributed and only the ontology, query and retrieval mechanisms need to be shared.

OWL- DL was used as foundation for ontology design which was used in the demonstration applications. Since the research was undertaken as part of a European project there were specific requirements from industry partners for data management. One request raised the subject of historical data management. This is a requirement because the occurrence some event in the past could be relevant to a current event. While some aspect of temporal data were addressed the particular problem of associating a current event with a previous event, such as measurement, was not addressed. This was scoping problem as the issue become too difficult to address in the context of all other developments. Therefore this is a matter that should be addressed through further work. Specifically the query functions that developed should cater for current conditions and conditions that occurred previously. The problem is in defining when a historic event is too old for consideration under current scope.

8.3 Further Work

The research detailed in this thesis was performed in conjunction with the InteGRail[†] project which aimed at addressing many of the issues associated with interoperability. This provided the opportunity to make contact with many railway stakeholders and to appreciate the complexity of the requirements and their concerns over integration issues. For example, one important issue that was uncovered as the project progressed was sensitivity over sharing data; some partners use measurement data for

[†] http://www.integrail.info/mis_stat.htm
The InteGRail project aims to create a holistic, coherent information system, integrating the major railway sub-systems, in order to achieve higher levels of performance of the railway system in terms of capacity, average speed and punctuality, safety and the optimised usage of resources.

commercial gain and were not agreeable to an open architecture for information interchange. This issue was not insurmountable in a technical sense but was not addressed in the work presented here. However, it remained as a challenge to investigating case studies which genuinely reflected the benefit of an ontology as a master meta-model for *types* of asset information. Cases where the value of capturing the context in information could be demonstrated were not fully investigated which, conversely, made it difficult to sell the concept to the stakeholders. Further work is required in a smaller scale project to analyse technically interesting integration opportunities more robustly.

The deployment of triple stores with overriding reasoning functions worked well as a prototype demonstration. However, the amount of data stored in the triple stores was not representative of that required by industry. In addition to this the type of queries required in an industrial setting are likely to be far more complex. A smaller scale case study, supported by a major industry stakeholder would enable the potential advantages of Semantic Web technology to be investigated more thoroughly. The support of a major stakeholder will ensure that access to the level of complexity required to demonstrate the key benefits of the technology is attained without the distraction of commercial concerns over data sensitivity.

The uptake of an integrating system based on a service oriented architecture and an ontology as the meta-model for information interchange depends upon buy in from industry stakeholders. Ontology based information systems are becoming more commonplace with enterprise level solutions such as Ontoprise[‡] targeting large

[‡] http://www.semafora-systems.com/en/products/ontobroker/

organisations. However, further work is required to build a foundation of knowledge within the railway network. Feasibility studies are required to create demonstration workbenches so that the appropriate individuals within an organisation can see the value of the ontology approach without the restrictions created by concerns over commercial risks.

The type of information considered throughout this thesis was, for the majority, static condition data. Therefore there is a requirement to address the needs of actors that require historical data to be available. It is easy to imagine, in a large, active system such as the railway domain, how the timing of one event with respect to other events is important. In such cases the description logics of an ontology may involve temporal patterns. This is an area that needs further investigation in order to ensure that legacy data can be utilised in the same was new event data.

Consideration was given to the wider information management and decision support requirements associated with condition monitoring. Figure 8.1 illustrates the ontology concepts derived to support a maintenance management application. This ontology model covers the wider context of information that may be associated with a condition monitoring event such as equipment and personnel requirements.

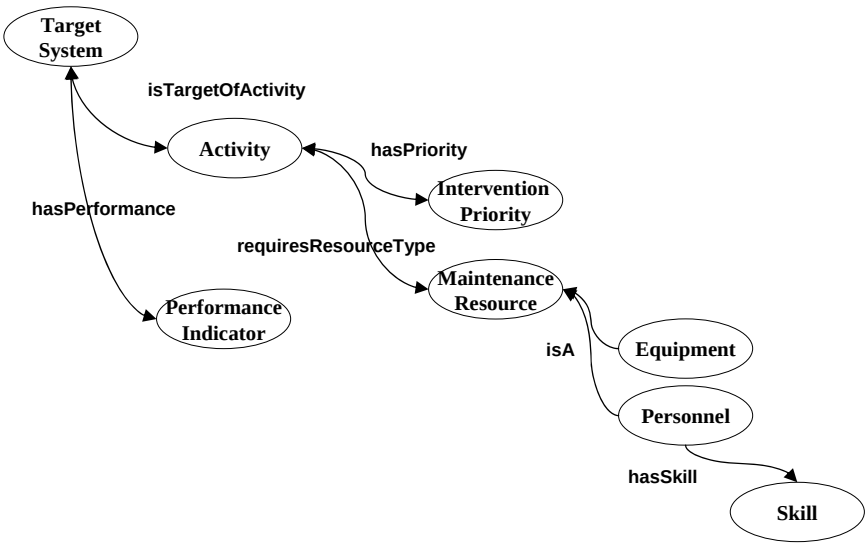


Figure 8.1 Ontology to support maintenance application

This work is outside of the scope of this thesis but serves to illustrate that information management and decision support potential extends beyond simple measurement context. The application of ontology can extend to wider resource management in decision support activities. The potential advantage of capturing context in semantics in enterprise knowledge management requires further exploration.

Appendix A

Journal Paper: Using non-monotonic reasoning to manage uncertainty in railway asset diagnostics

Expert Systems with Applications 37 (2010) 3616–3623

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Using non-monotonic reasoning to manage uncertainty in railway asset diagnostics

R. Lewis *, C. Roberts

Department of Electronic, Electrical and Computer Engineering, University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

ARTICLE INFO

Keywords:
Railway diagnostics
Ontology
Description logics
Non-monotonic reasoning

ABSTRACT

When a system instantaneous fails there is no time to perform preventative maintenance. In certain circumstances, where the system subsequently recovers without intervention, the system is left to run undiagnosed. Increasing the certainty of a particular diagnosis leads to a greater likelihood of the corrective action being carried out. Therefore, moving to a situation where available symptom and associated condition data is used to diagnose a failure is desirable. This paper proposes the implementation of a distributed network of semantic nodes that supports the inference of asset status. Querying is then performed to produce concrete facts regarding the status of the asset. The facts are used to support non-monotonic, probabilistic reasoning to increase the certainty that a particular symptom is the cause of the instantaneous failure.

Appendix B

Railway Domain Ontology – Core (RDF XML)

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY swrl "http://www.w3.org/2003/11/swrl#" >
   <!ENTITY swrlb "http://www.w3.org/2003/11/swrlb#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY protege "http://protege.stanford.edu/plugins/owl/protege#" >
   <!ENTITY xsp "http://www.owl-ontologies.com/2005/08/07/xsp.owl#" >
]>


<rdf:RDF xmlns="http://www.integrail.info/ont/SP3A.owl#"
     xml:base="http://www.integrail.info/ont/SP3A.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:swrl="http://www.w3.org/2003/11/swrl#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:xsp="http://www.owl-ontologies.com/2005/08/07/xsp.owl#"
     xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>
    <owl:Class rdf:ID="AccelerationStatus">
       <rdfs:subClassOf rdf:resource="#SystemStatus"/>
       <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
       <owl:disjointWith rdf:resource="#VehicleStatus"/>
       <owl:disjointWith rdf:resource="#WheelStatus"/>
       <owl:disjointWith rdf:resource="#DriveStatus"/>
       <owl:disjointWith rdf:resource="#BogieStatus"/>
       <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
       <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
       <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
       <owl:disjointWith rdf:resource="#TransmissionStatus"/>
       <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
       <owl:disjointWith rdf:resource="#WheelSetStatus"/>
       <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
       <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
       <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
       <owl:disjointWith rdf:resource="#CarBodyStatus"/>
       <owl:disjointWith rdf:resource="#TrackStatus"/>
       <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
       <owl:disjointWith rdf:resource="#JunctionStatus"/>
       <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
       <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
       <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
       <owl:disjointWith rdf:resource="#RideQualityStatus"/>
       <owl:disjointWith rdf:resource="#PointMachineStatus"/>
       <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
       <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
       <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
       <owl:disjointWith rdf:resource="#PointsStatus"/>
       <owl:disjointWith rdf:resource="#AxleStatus"/>
       <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
       <owl:disjointWith rdf:resource="#CarriageStatus"/>
       <owl:disjointWith rdf:resource="#RollingStatus"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="altitude">
       <rdf:type rdf:resource="&owl;FunctionalProperty"/>
       <rdfs:domain rdf:resource="#LocationCoordinates"/>
       <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="AuxiliarySystem">
       <rdfs:subClassOf rdf:resource="#BogieComponent"/>
       <owl:disjointWith rdf:resource="#PrimarySuspension"/>
       <owl:disjointWith rdf:resource="#SecondarySuspension"/>
       <owl:disjointWith rdf:resource="#MotorInterface"/>
       <owl:disjointWith rdf:resource="#CoolingSystem"/>
```

```
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
  </owl:Class>
  <owl:Class rdf:ID="AuxiliarySystemStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
  </owl:Class>
  <owl:Class rdf:ID="Axle">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
  </owl:Class>
  <owl:Class rdf:ID="AxleBearingStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
```

```
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="AxleStatus">
        <rdfs:subClassOf rdf:resource="#SystemStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="Bearing">
        <rdfs:subClassOf rdf:resource="#BogieComponent"/>
        <owl:disjointWith rdf:resource="#GearBoxInterface"/>
        <owl:disjointWith rdf:resource="#BogieLinkage"/>
        <owl:disjointWith rdf:resource="#BearingBox"/>
```

```
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
  </owl:Class>
  <owl:Class rdf:ID="BearingAssembly">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="BearingBox">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
  </owl:Class>
  <owl:Class rdf:ID="Bogie">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
```

```
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
  </owl:Class>
  <owl:Class rdf:ID="BogieComponent">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="BogieFrame">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="BogieLinkage">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="BogieStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
```

```
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="BrakeDiskInterface">
        <rdfs:subClassOf rdf:resource="#BogieComponent"/>
        <owl:disjointWith rdf:resource="#BearingBox"/>
        <owl:disjointWith rdf:resource="#PrimarySuspension"/>
        <owl:disjointWith rdf:resource="#Bearing"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
        <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
        <owl:disjointWith rdf:resource="#BogieFrame"/>
        <owl:disjointWith rdf:resource="#Wheel"/>
        <owl:disjointWith rdf:resource="#BearingAssembly"/>
        <owl:disjointWith rdf:resource="#CoolingSystem"/>
        <owl:disjointWith rdf:resource="#Transmission"/>
        <owl:disjointWith rdf:resource="#BogieLinkage"/>
        <owl:disjointWith rdf:resource="#WheelSetSystem"/>
        <owl:disjointWith rdf:resource="#MotorInterface"/>
        <owl:disjointWith rdf:resource="#WheelSet"/>
        <owl:disjointWith rdf:resource="#Axle"/>
        <owl:disjointWith rdf:resource="#SecondarySuspension"/>
        <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    </owl:Class>
    <owl:Class rdf:ID="BrakingSystem">
        <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
        <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
        <owl:disjointWith rdf:resource="#PowerSystem"/>
        <owl:disjointWith rdf:resource="#DoorSystem"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
        <owl:disjointWith rdf:resource="#TrainConsist"/>
        <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
        <owl:disjointWith rdf:resource="#Window"/>
        <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
        <owl:disjointWith rdf:resource="#CarBody"/>
        <owl:disjointWith rdf:resource="#PropulsionSystem"/>
        <owl:disjointWith rdf:resource="#Vehicle"/>
        <owl:disjointWith rdf:resource="#Bogie"/>
        <owl:disjointWith rdf:resource="#BogieComponent"/>
    </owl:Class>
    <owl:Class rdf:ID="BrakingSystemComponent">
        <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
        <owl:disjointWith rdf:resource="#BogieComponent"/>
        <owl:disjointWith rdf:resource="#PropulsionSystem"/>
        <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
        <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
```

```
      <owl:disjointWith rdf:resource="#TrainConsist"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
      <owl:disjointWith rdf:resource="#CarBody"/>
      <owl:disjointWith rdf:resource="#DoorSystem"/>
      <owl:disjointWith rdf:resource="#BrakingSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
      <owl:disjointWith rdf:resource="#Window"/>
      <owl:disjointWith rdf:resource="#Vehicle"/>
      <owl:disjointWith rdf:resource="#Bogie"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
      <owl:disjointWith rdf:resource="#PowerSystem"/>
    </owl:Class>
    <owl:Class rdf:ID="BrakingSystemStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="BufferStop">
      <rdfs:subClassOf rdf:resource="#TrackNetworkNode"/>
      <owl:disjointWith rdf:resource="#Platform"/>
      <owl:disjointWith rdf:resource="#SandDrag"/>
      <owl:disjointWith rdf:resource="#Signal"/>
      <owl:disjointWith rdf:resource="#TerraIncognita"/>
    </owl:Class>
    <owl:Class rdf:ID="CarBody">
      <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
      <owl:disjointWith rdf:resource="#DoorSystem"/>
      <owl:disjointWith rdf:resource="#BogieComponent"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
      <owl:disjointWith rdf:resource="#TrainConsist"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
      <owl:disjointWith rdf:resource="#Window"/>
      <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
      <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
      <owl:disjointWith rdf:resource="#PowerSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
      <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
      <owl:disjointWith rdf:resource="#BrakingSystem"/>
      <owl:disjointWith rdf:resource="#Bogie"/>
```

```
      <owl:disjointWith rdf:resource="#Vehicle"/>
      <owl:disjointWith rdf:resource="#PropulsionSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    </owl:Class>
    <owl:Class rdf:ID="CarBodyStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="CarriageStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
```

```
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="causedBySymptom">
    <rdfs:domain rdf:resource="#Fault"/>
    <rdfs:range rdf:resource="#Symptom"/>
    <owl:inverseOf rdf:resource="#refersToFault"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Chains">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#RailwayLocationCoordinates"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Class4FreightTrainConsist">
    <rdfs:subClassOf rdf:resource="#FreightTrainConsist"/>
    <owl:disjointWith rdf:resource="#StandardFreightTrainConsist"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="consist_ID">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#TrainConsist"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="CoolingSystem">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
  </owl:Class>
  <owl:Class rdf:ID="CoolingSystemStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
```

```
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
</owl:Class>
<owl:Class rdf:ID="CriticalFault">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#causedBySymptom"/>
                    <owl:someValuesFrom>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#hasEventLevel"/>
                            <owl:someValuesFrom rdf:resource="#High"/>
                        </owl:Restriction>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Class rdf:about="#Fault"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="CriticalStatus">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isStatusOf"/>
                    <owl:someValuesFrom>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#hasObservation"/>
                            <owl:someValuesFrom>
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasSymptom"/>
                                    <owl:someValuesFrom>
                                        <owl:Restriction>
                                            <owl:onProperty rdf:resource="#refersToFault"/>
                                            <owl:allValuesFrom
rdf:resource="#CriticalFault"/>
                                        </owl:Restriction>
                                    </owl:someValuesFrom>
                                </owl:Restriction>
                            </owl:someValuesFrom>
                        </owl:Restriction>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isStatusOf"/>
                    <owl:someValuesFrom>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#hasObservation"/>
                            <owl:someValuesFrom>
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasSymptom"/>
                                    <owl:someValuesFrom>
                                        <owl:Restriction>
                                            <owl:onProperty rdf:resource="#refersToFault"/>
                                            <owl:someValuesFrom
rdf:resource="#CriticalFault"/>
                                        </owl:Restriction>
                                    </owl:someValuesFrom>
                                </owl:Restriction>
                            </owl:someValuesFrom>
                        </owl:Restriction>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Class rdf:about="#SystemStatus"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="Crossing">
```

```
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
              <owl:Class rdf:about="#DiamondCrossing"/>
              <owl:Class rdf:about="#ObliqueCrossing"/>
              <owl:Class rdf:about="#Right-AngleCrossing"/>
            </owl:unionOf>
          </owl:Class>
          <owl:Class rdf:about="#TrackNetworkNode"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
  </owl:Class>
  <owl:Class rdf:ID="Decreasing">
    <rdfs:subClassOf rdf:resource="#Trend"/>
    <owl:disjointWith rdf:resource="#Increasing"/>
    <owl:disjointWith rdf:resource="#Steady"/>
  </owl:Class>
  <Decreasing rdf:ID="decreasing"/>
  <owl:Class rdf:ID="Depot">
    <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
    <owl:disjointWith rdf:resource="#FreightTerminal"/>
    <owl:disjointWith rdf:resource="#Junction"/>
    <owl:disjointWith rdf:resource="#RegionalBoundary"/>
    <owl:disjointWith rdf:resource="#Siding"/>
    <owl:disjointWith rdf:resource="#Station"/>
    <owl:disjointWith rdf:resource="#Tunnel"/>
  </owl:Class>
  <owl:Class rdf:ID="DiamondCrossing">
    <rdfs:subClassOf rdf:resource="#Crossing"/>
    <owl:disjointWith rdf:resource="#ObliqueCrossing"/>
    <owl:disjointWith rdf:resource="#Right-AngleCrossing"/>
  </owl:Class>
  <owl:Class rdf:ID="DieselMotorBogie">
    <rdfs:subClassOf rdf:resource="#MotorBogie"/>
    <owl:disjointWith rdf:resource="#ElectricMotorBogie"/>
  </owl:Class>
  <owl:Class rdf:ID="Direction">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Forward"/>
          <owl:Class rdf:about="#Reverse"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#ValuePartition"/>
  </owl:Class>
  <owl:Class rdf:ID="DoorController">
    <rdfs:subClassOf rdf:resource="#DoorSystemComponent"/>
  </owl:Class>
  <owl:Class rdf:ID="DoorSystem">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
```

```
    </owl:Class>
    <owl:Class rdf:ID="DoorSystemComponent">
        <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
        <owl:disjointWith rdf:resource="#Vehicle"/>
        <owl:disjointWith rdf:resource="#Window"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
        <owl:disjointWith rdf:resource="#PowerSystem"/>
        <owl:disjointWith rdf:resource="#TrainConsist"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
        <owl:disjointWith rdf:resource="#BrakingSystem"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
        <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
        <owl:disjointWith rdf:resource="#DoorSystem"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
        <owl:disjointWith rdf:resource="#PropulsionSystem"/>
        <owl:disjointWith rdf:resource="#BogieComponent"/>
        <owl:disjointWith rdf:resource="#Bogie"/>
        <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
        <owl:disjointWith rdf:resource="#CarBody"/>
    </owl:Class>
    <owl:Class rdf:ID="DoorSystemStatus">
        <rdfs:subClassOf rdf:resource="#SystemStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="DriveStatus">
        <rdfs:subClassOf rdf:resource="#SystemStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
```

```
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="ElectricalAsset">
        <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
        <owl:disjointWith rdf:resource="#MechanicalAsset"/>
        <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
        <owl:disjointWith rdf:resource="#PointMachine"/>
        <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
        <owl:disjointWith rdf:resource="#PermanentWayObject"/>
        <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
    </owl:Class>
    <owl:Class rdf:ID="ElectricDoorSystem">
        <rdfs:subClassOf rdf:resource="#DoorSystem"/>
        <owl:disjointWith rdf:resource="#ManualDoorSystem"/>
        <owl:disjointWith rdf:resource="#PneumaticDoorSystem"/>
    </owl:Class>
    <owl:Class rdf:ID="ElectricMotorBogie">
        <rdfs:subClassOf rdf:resource="#MotorBogie"/>
        <owl:disjointWith rdf:resource="#DieselMotorBogie"/>
</owl:Class>
<owl:Class rdf:ID="ElectromechanicalAsset">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
    <owl:disjointWith rdf:resource="#ElectricalAsset"/>
    <owl:disjointWith rdf:resource="#MechanicalAsset"/>
    <owl:disjointWith rdf:resource="#PointMachine"/>
    <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
    <owl:disjointWith rdf:resource="#PermanentWayObject"/>
    <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="ELR">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#RailwayLocationCoordinates"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="EmergencyBrakeEquipment">
    <rdfs:subClassOf rdf:resource="#BrakingSystem"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="endsAt">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#NetworkTopologyEdge"/>
                <owl:Class rdf:about="#RouteNetworkEdge"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#NetworkTopologyNode"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="EventLevel">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#High"/>
                <owl:Class rdf:about="#Low"/>
                <owl:Class rdf:about="#Med"/>
```

```
            </owl:unionOf>
         </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#ValuePartition"/>
   </owl:Class>
   <owl:Class rdf:ID="ExternalDoorSystem">
      <rdfs:subClassOf rdf:resource="#DoorSystem"/>
      <owl:disjointWith rdf:resource="#InternalDoorSystem"/>
   </owl:Class>
   <owl:Class rdf:ID="Fault">
      <owl:equivalentClass>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <owl:Class rdf:about="#CriticalFault"/>
               <owl:Class rdf:about="#IncipientFault"/>
               <owl:Class rdf:about="#NonCriticalFault"/>
            </owl:unionOf>
         </owl:Class>
      </owl:equivalentClass>
      <owl:disjointWith rdf:resource="#Symptom"/>
      <owl:disjointWith rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
      <owl:disjointWith rdf:resource="#Route"/>
      <owl:disjointWith rdf:resource="#Observation"/>
      <owl:disjointWith rdf:resource="#ObservationData"/>
      <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
      <owl:disjointWith rdf:resource="#Observer"/>
      <owl:disjointWith rdf:resource="#ValuePartition"/>
      <owl:disjointWith rdf:resource="#TargetSystem"/>
      <owl:disjointWith rdf:resource="#LocationCoordinates"/>
   </owl:Class>
   <owl:Class rdf:ID="FireDetectionSystem">
      <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
      <owl:disjointWith rdf:resource="#CarBody"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
      <owl:disjointWith rdf:resource="#BogieComponent"/>
      <owl:disjointWith rdf:resource="#Vehicle"/>
      <owl:disjointWith rdf:resource="#Window"/>
      <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
      <owl:disjointWith rdf:resource="#PropulsionSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
      <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
      <owl:disjointWith rdf:resource="#Bogie"/>
      <owl:disjointWith rdf:resource="#TrainConsist"/>
      <owl:disjointWith rdf:resource="#DoorSystem"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
      <owl:disjointWith rdf:resource="#BrakingSystem"/>
      <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
      <owl:disjointWith rdf:resource="#PowerSystem"/>
   </owl:Class>
   <owl:Class rdf:ID="FireDetectionSystemComponent">
      <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
      <owl:disjointWith rdf:resource="#BrakingSystem"/>
      <owl:disjointWith rdf:resource="#Vehicle"/>
      <owl:disjointWith rdf:resource="#Window"/>
      <owl:disjointWith rdf:resource="#PowerSystem"/>
      <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
      <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
      <owl:disjointWith rdf:resource="#CarBody"/>
      <owl:disjointWith rdf:resource="#PropulsionSystem"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
      <owl:disjointWith rdf:resource="#DoorSystem"/>
      <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
      <owl:disjointWith rdf:resource="#BogieComponent"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
      <owl:disjointWith rdf:resource="#TrainConsist"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
      <owl:disjointWith rdf:resource="#Bogie"/>
   </owl:Class>
   <owl:Class rdf:ID="FireDetectionSystemStatus">
```

```
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
   </owl:Class>
   <owl:Class rdf:ID="Forward">
      <rdfs:subClassOf rdf:resource="#Direction"/>
      <owl:disjointWith rdf:resource="#Reverse"/>
   </owl:Class>
   <owl:Class rdf:ID="FreightCar">
      <rdfs:subClassOf rdf:resource="#Railcar"/>
      <owl:disjointWith rdf:resource="#PassengerCar"/>
   </owl:Class>
   <owl:Class rdf:ID="FreightTerminal">
      <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
      <owl:disjointWith rdf:resource="#Depot"/>
      <owl:disjointWith rdf:resource="#Junction"/>
      <owl:disjointWith rdf:resource="#RegionalBoundary"/>
      <owl:disjointWith rdf:resource="#Siding"/>
      <owl:disjointWith rdf:resource="#Station"/>
      <owl:disjointWith rdf:resource="#Tunnel"/>
   </owl:Class>
   <owl:Class rdf:ID="FreightTrainConsist">
      <rdfs:subClassOf rdf:resource="#TrainConsist"/>
      <owl:disjointWith rdf:resource="#PassengerTrainConsist"/>
   </owl:Class>
   <Forward rdf:ID="fwd"/>
   <owl:Class rdf:ID="GearBoxInterface">
      <rdfs:subClassOf rdf:resource="#BogieComponent"/>
      <owl:disjointWith rdf:resource="#PrimarySuspension"/>
      <owl:disjointWith rdf:resource="#CoolingSystem"/>
      <owl:disjointWith rdf:resource="#WheelSetSystem"/>
      <owl:disjointWith rdf:resource="#Wheel"/>
      <owl:disjointWith rdf:resource="#BearingBox"/>
      <owl:disjointWith rdf:resource="#Axle"/>
      <owl:disjointWith rdf:resource="#BogieLinkage"/>
      <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
      <owl:disjointWith rdf:resource="#Bearing"/>
      <owl:disjointWith rdf:resource="#WheelSet"/>
      <owl:disjointWith rdf:resource="#BogieFrame"/>
      <owl:disjointWith rdf:resource="#SecondarySuspension"/>
      <owl:disjointWith rdf:resource="#BearingAssembly"/>
      <owl:disjointWith rdf:resource="#Transmission"/>
```

```
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="hasAsset">
    <rdfs:domain rdf:resource="#NetworkTopologyNode"/>
    <rdfs:range rdf:resource="#NetworkTopologyAsset"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDirection">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="#Direction"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasDirectPart">
    <rdfs:subPropertyOf rdf:resource="#hasPart"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasEventLevel">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Symptom"/>
    <rdfs:range rdf:resource="#EventLevel"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFeature">
    <rdfs:domain rdf:resource="#NetworkTopologyEdge"/>
    <rdfs:range rdf:resource="#NetworkTopologyNode"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFirstConsist">
    <rdfs:domain rdf:resource="#OperationalTrain"/>
    <rdfs:range rdf:resource="#TrainConsist"/>
    <owl:inverseOf rdf:resource="#isOfOpVehicle"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFirstRouteSegment">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Route"/>
    <rdfs:range rdf:resource="#RouteSegment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasFirstVehicle">
    <rdfs:domain rdf:resource="#TrainConsist"/>
    <rdfs:range rdf:resource="#Vehicle"/>
    <owl:inverseOf rdf:resource="#isFirstVehicleOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasLocation">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Observer"/>
                <owl:Class rdf:about="#TargetSystem"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#Location"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextConsist">
    <rdfs:domain rdf:resource="#TrainConsist"/>
    <rdfs:range rdf:resource="#TrainConsist"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextDirectRouteSegment">
    <rdfs:subPropertyOf rdf:resource="#hasNextRouteSegment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextObservation">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Observation"/>
    <rdfs:range rdf:resource="#Observation"/>
    <owl:inverseOf rdf:resource="#hasPreviousObservation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextRouteSegment">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="#RouteSegment"/>
    <rdfs:range rdf:resource="#RouteSegment"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasNextVehicle">
```

```
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="#Vehicle"/>
    <owl:inverseOf rdf:resource="#hasPreviousVehicle"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasObservation">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdfs:domain rdf:resource="#TargetSystem"/>
    <rdfs:range rdf:resource="#Observation"/>
    <owl:inverseOf rdf:resource="#refersToSystem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasObservationData">
    <rdfs:domain rdf:resource="#Observation"/>
    <rdfs:range rdf:resource="#ObservationData"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPart">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="#TargetSystem"/>
    <rdfs:range rdf:resource="#TargetSystem"/>
    <owl:inverseOf rdf:resource="#isPartOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPreviousObservation">
    <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Observation"/>
    <rdfs:range rdf:resource="#Observation"/>
    <owl:inverseOf rdf:resource="#hasNextObservation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasPreviousVehicle">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="#Vehicle"/>
    <owl:inverseOf rdf:resource="#hasNextVehicle"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasRunningLine">
    <rdfs:domain rdf:resource="#LineOfRoute"/>
    <rdfs:range rdf:resource="#Track"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSide">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#RollingStockObject"/>
    <rdfs:range rdf:resource="#Side"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasStatus">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#TargetSystem"/>
    <rdfs:range rdf:resource="#SystemStatus"/>
    <owl:inverseOf rdf:resource="#isStatusOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasSymptom">
    <rdfs:domain rdf:resource="#Observation"/>
    <rdfs:range rdf:resource="#Symptom"/>
    <owl:inverseOf rdf:resource="#refersToObservation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hasTrend">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Symptom"/>
    <rdfs:range rdf:resource="#Trend"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="hasValue">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Observation"/>
    <rdfs:range rdf:resource="&xsd;int"/>
</owl:DatatypeProperty>
<owl:ObjectProperty rdf:ID="hasVehicle">
    <rdfs:domain rdf:resource="#TrainConsist"/>
    <rdfs:range rdf:resource="#Vehicle"/>
    <owl:inverseOf rdf:resource="#isVehicleOf"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="High">
    <rdfs:subClassOf rdf:resource="#EventLevel"/>
    <owl:disjointWith rdf:resource="#Low"/>
```

```
    <owl:disjointWith rdf:resource="#Med"/>
</owl:Class>
<High rdf:ID="high"/>
<owl:Class rdf:ID="HighSpeedPassengerTrainConsist">
    <rdfs:subClassOf rdf:resource="#PassengerTrainConsist"/>
    <owl:disjointWith rdf:resource="#StandardPassengerTrainConsist"/>
</owl:Class>
<owl:Class rdf:ID="HumanObserver">
    <rdfs:subClassOf rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#MeasurementSystem"/>
</owl:Class>
<owl:Class rdf:ID="IncipientFault">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#causedBySymptom"/>
                    <owl:someValuesFrom>
                        <owl:Class>
                            <owl:unionOf rdf:parseType="Collection">
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasTrend"/>
                                    <owl:someValuesFrom rdf:resource="#Decreasing"/>
                                </owl:Restriction>
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasTrend"/>
                                    <owl:someValuesFrom rdf:resource="#Increasing"/>
                                </owl:Restriction>
                            </owl:unionOf>
                        </owl:Class>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Class rdf:about="#Fault"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="IncipientStatus">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isStatusOf"/>
                    <owl:someValuesFrom>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#hasObservation"/>
                            <owl:someValuesFrom>
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasSymptom"/>
                                    <owl:someValuesFrom>
                                        <owl:Restriction>
                                            <owl:onProperty rdf:resource="#refersToFault"/>
                                            <owl:allValuesFrom
rdf:resource="#IncipientFault"/>
                                        </owl:Restriction>
                                    </owl:someValuesFrom>
                                </owl:Restriction>
                            </owl:someValuesFrom>
                        </owl:Restriction>
                    </owl:someValuesFrom>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isStatusOf"/>
                    <owl:someValuesFrom>
                        <owl:Restriction>
                            <owl:onProperty rdf:resource="#hasObservation"/>
                            <owl:someValuesFrom>
                                <owl:Restriction>
                                    <owl:onProperty rdf:resource="#hasSymptom"/>
                                    <owl:someValuesFrom>
                                        <owl:Restriction>
                                            <owl:onProperty rdf:resource="#refersToFault"/>
```

```
                                    <owl:someValuesFrom
rdf:resource="#IncipientFault"/>
                                </owl:Restriction>
                            </owl:someValuesFrom>
                        </owl:Restriction>
                    </owl:someValuesFrom>
                </owl:Restriction>
            </owl:someValuesFrom>
        </owl:Restriction>
        <owl:Class rdf:about="#SystemStatus"/>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
 </owl:Class>
 <owl:Class rdf:ID="Increasing">
  <rdfs:subClassOf rdf:resource="#Trend"/>
  <owl:disjointWith rdf:resource="#Steady"/>
  <owl:disjointWith rdf:resource="#Decreasing"/>
 </owl:Class>
 <Increasing rdf:ID="increasing"/>
 <owl:Class rdf:ID="InternalDoorSystem">
  <rdfs:subClassOf rdf:resource="#DoorSystem"/>
  <owl:disjointWith rdf:resource="#ExternalDoorSystem"/>
 </owl:Class>
 <owl:ObjectProperty rdf:ID="isComprisedOf">
  <rdfs:domain rdf:resource="#NetworkTopologyElement"/>
  <rdfs:range rdf:resource="#NetworkTopologyElement"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isFirstVehicleOf">
  <rdfs:domain rdf:resource="#Vehicle"/>
  <rdfs:range rdf:resource="#TrainConsist"/>
  <owl:inverseOf rdf:resource="#hasFirstVehicle"/>
 </owl:ObjectProperty>
 <owl:DatatypeProperty rdf:ID="isLeadingConsist">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#TrainConsist"/>
  <rdfs:range rdf:resource="&xsd;boolean"/>
 </owl:DatatypeProperty>
 <owl:ObjectProperty rdf:ID="isLocatedAt">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:domain rdf:resource="#Observer"/>
  <rdfs:range rdf:resource="#TargetSystem"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isMappedToLineOfRoute">
  <rdfs:domain rdf:resource="#RouteSegment"/>
  <rdfs:range rdf:resource="#Track"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isMostRecent">
  <rdf:type rdf:resource="&owl;AnnotationProperty"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isObservationFrom">
  <rdfs:domain rdf:resource="#Observation"/>
  <rdfs:range rdf:resource="#Observer"/>
  <owl:inverseOf rdf:resource="#ownsObservation"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isOfOpVehicle">
  <rdfs:domain rdf:resource="#TrainConsist"/>
  <rdfs:range rdf:resource="#OperationalTrain"/>
  <owl:inverseOf rdf:resource="#hasFirstConsist"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isPartOf">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdfs:domain rdf:resource="#TargetSystem"/>
  <rdfs:range rdf:resource="#TargetSystem"/>
  <owl:inverseOf rdf:resource="#hasPart"/>
 </owl:ObjectProperty>
 <owl:ObjectProperty rdf:ID="isStatusOf">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
  <rdfs:domain rdf:resource="#SystemStatus"/>
  <rdfs:range rdf:resource="#TargetSystem"/>
  <owl:inverseOf rdf:resource="#hasStatus"/>
 </owl:ObjectProperty>
```

```
    <owl:ObjectProperty rdf:ID="isVehicleOf">
      <rdfs:domain rdf:resource="#Vehicle"/>
      <rdfs:range rdf:resource="#TrainConsist"/>
      <owl:inverseOf rdf:resource="#hasVehicle"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Junction">
      <owl:equivalentClass>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isComprisedOf"/>
              <owl:allValuesFrom rdf:resource="#Switch"/>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isComprisedOf"/>
              <owl:minCardinality
rdf:datatype="&xsd;int">1</owl:minCardinality>
            </owl:Restriction>
            <owl:Class rdf:about="#LineNetworkNode"/>
          </owl:intersectionOf>
        </owl:Class>
      </owl:equivalentClass>
      <owl:disjointWith rdf:resource="#FreightTerminal"/>
      <owl:disjointWith rdf:resource="#RegionalBoundary"/>
      <owl:disjointWith rdf:resource="#Siding"/>
      <owl:disjointWith rdf:resource="#Station"/>
      <owl:disjointWith rdf:resource="#Tunnel"/>
      <owl:disjointWith rdf:resource="#Depot"/>
    </owl:Class>
    <owl:Class rdf:ID="JunctionStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="latitude">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="#LocationCoordinates"/>
      <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
    <owl:Class rdf:ID="Left">
      <rdfs:subClassOf rdf:resource="#Side"/>
      <owl:disjointWith rdf:resource="#Right"/>
```

```
</owl:Class>
<Left rdf:ID="left"/>
<owl:DatatypeProperty rdf:ID="lineID">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#LineNetworkEdge"/>
   <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="LineNetworkEdge">
   <rdfs:subClassOf rdf:resource="#NetworkTopologyEdge"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#startsAt"/>
         <owl:someValuesFrom rdf:resource="#LineNetworkNode"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#startsAt"/>
         <owl:allValuesFrom rdf:resource="#LineNetworkNode"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#endsAt"/>
         <owl:someValuesFrom rdf:resource="#LineNetworkNode"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#endsAt"/>
         <owl:allValuesFrom rdf:resource="#LineNetworkNode"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <owl:disjointWith rdf:resource="#TrackNetworkEdge"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="LineNetworkEdgeHasTrackNetworkEdge">
   <rdfs:domain rdf:resource="#LineNetworkEdge"/>
   <rdfs:range rdf:resource="#TrackNetworkEdge"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="LineNetworkNode">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <owl:Class>
               <owl:unionOf rdf:parseType="Collection">
                  <owl:Class rdf:about="#Depot"/>
                  <owl:Class rdf:about="#FreightTerminal"/>
                  <owl:Class rdf:about="#Junction"/>
                  <owl:Class rdf:about="#RegionalBoundary"/>
                  <owl:Class rdf:about="#Siding"/>
                  <owl:Class rdf:about="#Station"/>
                  <owl:Class rdf:about="#Tunnel"/>
               </owl:unionOf>
            </owl:Class>
            <owl:Class rdf:about="#NetworkTopologyNode"/>
         </owl:intersectionOf>
      </owl:Class>
   </owl:equivalentClass>
   <owl:disjointWith rdf:resource="#TrackNetworkNode"/>
</owl:Class>
<owl:Class rdf:ID="LineOfRoute">
   <rdfs:subClassOf rdf:resource="#LineNetworkEdge"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasRunningLine"/>
         <owl:someValuesFrom rdf:resource="#Track"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasRunningLine"/>
         <owl:allValuesFrom rdf:resource="#Track"/>
```

```
            </owl:Restriction>
          </rdfs:subClassOf>
        </owl:Class>
        <owl:Class rdf:ID="LineOfRouteStatus">
          <rdfs:subClassOf rdf:resource="#SystemStatus"/>
          <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
          <owl:disjointWith rdf:resource="#JunctionStatus"/>
          <owl:disjointWith rdf:resource="#VehicleStatus"/>
          <owl:disjointWith rdf:resource="#BogieStatus"/>
          <owl:disjointWith rdf:resource="#CarBodyStatus"/>
          <owl:disjointWith rdf:resource="#TransmissionStatus"/>
          <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
          <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
          <owl:disjointWith
    rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
          <owl:disjointWith rdf:resource="#PointsStatus"/>
          <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
          <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
          <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
          <owl:disjointWith rdf:resource="#AccelerationStatus"/>
          <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
          <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
          <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
          <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
          <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
          <owl:disjointWith rdf:resource="#TrackStatus"/>
          <owl:disjointWith rdf:resource="#WheelStatus"/>
          <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
          <owl:disjointWith rdf:resource="#DriveStatus"/>
          <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
          <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
          <owl:disjointWith rdf:resource="#AxleStatus"/>
          <owl:disjointWith rdf:resource="#CarriageStatus"/>
          <owl:disjointWith rdf:resource="#WheelSetStatus"/>
          <owl:disjointWith rdf:resource="#RollingStatus"/>
          <owl:disjointWith rdf:resource="#RideQualityStatus"/>
          <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        </owl:Class>
        <owl:Class rdf:ID="LinesideMeasurementSystem">
          <owl:equivalentClass>
            <owl:Class>
              <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#isLocatedAt"/>
                  <owl:allValuesFrom rdf:resource="#NetworkTopologyElement"/>
                </owl:Restriction>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#isLocatedAt"/>
                  <owl:someValuesFrom
    rdf:resource="#NetworkTopologyElement"/>
                </owl:Restriction>
                <owl:Class rdf:about="#MeasurementSystem"/>
              </owl:intersectionOf>
            </owl:Class>
          </owl:equivalentClass>
        </owl:Class>
        <owl:Class rdf:ID="Location">
          <owl:equivalentClass>
            <owl:Class>
              <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#LocationCoordinates"/>
                <owl:Class rdf:about="#RailwayLocationCoordinates"/>
              </owl:unionOf>
            </owl:Class>
          </owl:equivalentClass>
        </owl:Class>
        <owl:Class rdf:ID="LocationCoordinates">
          <rdfs:subClassOf rdf:resource="#Location"/>
          <owl:disjointWith rdf:resource="#Fault"/>
          <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
          <owl:disjointWith rdf:resource="#TargetSystem"/>
          <owl:disjointWith rdf:resource="#SystemStatus"/>
```

```
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#Symptom"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
  </owl:Class>
  <owl:Class rdf:ID="Locomotive">
    <rdfs:subClassOf rdf:resource="#SelfPoweredVehicle"/>
    <owl:disjointWith rdf:resource="#SelfPoweredRailcar"/>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="longitute">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#LocationCoordinates"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="Low">
    <rdfs:subClassOf rdf:resource="#EventLevel"/>
    <owl:disjointWith rdf:resource="#Med"/>
    <owl:disjointWith rdf:resource="#High"/>
  </owl:Class>
  <Low rdf:ID="low"/>
  <owl:Class rdf:ID="ManualDoorSystem">
    <rdfs:subClassOf rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#ElectricDoorSystem"/>
    <owl:disjointWith rdf:resource="#PneumaticDoorSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="MeasurementSystem">
    <rdfs:subClassOf rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#HumanObserver"/>
  </owl:Class>
  <owl:Class rdf:ID="MechanicalAsset">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
    <owl:disjointWith rdf:resource="#ElectricalAsset"/>
    <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
    <owl:disjointWith rdf:resource="#PointMachine"/>
    <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
    <owl:disjointWith rdf:resource="#PermanentWayObject"/>
    <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
  </owl:Class>
  <owl:Class rdf:ID="Med">
    <rdfs:subClassOf rdf:resource="#EventLevel"/>
    <owl:disjointWith rdf:resource="#Low"/>
    <owl:disjointWith rdf:resource="#High"/>
  </owl:Class>
  <Med rdf:ID="med"/>
  <owl:DatatypeProperty rdf:ID="Miles">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#RailwayLocationCoordinates"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="MotorBogie">
    <rdfs:subClassOf rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#TrailerBogie"/>
  </owl:Class>
  <owl:Class rdf:ID="MotorInterface">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
```

```
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
</owl:Class>
<owl:Class rdf:ID="NetworkTopologyAsset">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyElement"/>
    <owl:disjointWith rdf:resource="#NetworkTopologyEdge"/>
</owl:Class>
<owl:Class rdf:ID="NetworkTopologyEdge">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#endsAt"/>
                    <owl:allValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#endsAt"/>
                    <owl:someValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasFeature"/>
                    <owl:allValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#hasFeature"/>
                    <owl:someValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#LineNetworkEdge"/>
                        <owl:Class rdf:about="#TrackNetworkEdge"/>
                    </owl:unionOf>
                </owl:Class>
                <owl:Class rdf:about="#NetworkTopologyElement"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#startsAt"/>
                    <owl:allValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#startsAt"/>
                    <owl:someValuesFrom rdf:resource="#NetworkTopologyNode"/>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#NetworkTopologyNode"/>
    <owl:disjointWith rdf:resource="#NetworkTopologyAsset"/>
</owl:Class>
<owl:Class rdf:ID="NetworkTopologyElement">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#NetworkTopologyAsset"/>
                        <owl:Class rdf:about="#NetworkTopologyEdge"/>
                        <owl:Class rdf:about="#NetworkTopologyNode"/>
                    </owl:unionOf>
                </owl:Class>
                <owl:Class rdf:about="#TargetSystem"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#RollingStockObject"/>
</owl:Class>
<owl:Class rdf:ID="NetworkTopologyNode">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#NetworkTopologyElement"/>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
```

```
                <owl:Class rdf:about="#LineNetworkNode"/>
                <owl:Class rdf:about="#TrackNetworkNode"/>
              </owl:unionOf>
            </owl:Class>
          </owl:intersectionOf>
        </owl:Class>
      </owl:equivalentClass>
      <owl:disjointWith rdf:resource="#NetworkTopologyEdge"/>
    </owl:Class>
    <owl:Class rdf:ID="NonCriticalFault">
      <owl:equivalentClass>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#causedBySymptom"/>
              <owl:someValuesFrom>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasEventLevel"/>
                  <owl:allValuesFrom>
                    <owl:Class>
                      <owl:complementOf rdf:resource="#High"/>
                    </owl:Class>
                  </owl:allValuesFrom>
                </owl:Restriction>
              </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#causedBySymptom"/>
              <owl:someValuesFrom>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasEventLevel"/>
                  <owl:someValuesFrom>
                    <owl:Class>
                      <owl:complementOf rdf:resource="#High"/>
                    </owl:Class>
                  </owl:someValuesFrom>
                </owl:Restriction>
              </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isStatusOf"/>
              <owl:someValuesFrom>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasObservation"/>
                  <owl:someValuesFrom>
                    <owl:Restriction>
                      <owl:onProperty rdf:resource="#hasSymptom"/>
                      <owl:someValuesFrom>
                        <owl:Restriction>
                          <owl:onProperty rdf:resource="#refersToFault"/>
                          <owl:allValuesFrom
rdf:resource="#NonCriticalFault"/>
                        </owl:Restriction>
                      </owl:someValuesFrom>
                    </owl:Restriction>
                  </owl:someValuesFrom>
                </owl:Restriction>
              </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isStatusOf"/>
              <owl:someValuesFrom>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasObservation"/>
                  <owl:someValuesFrom>
                    <owl:Restriction>
                      <owl:onProperty rdf:resource="#hasSymptom"/>
                      <owl:someValuesFrom>
                        <owl:Restriction>
                          <owl:onProperty rdf:resource="#refersToFault"/>
                          <owl:allValuesFrom
rdf:resource="#NonCriticalFault"/>
                        </owl:Restriction>
                      </owl:someValuesFrom>
                    </owl:Restriction>
                  </owl:someValuesFrom>
                </owl:Restriction>
              </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Class rdf:about="#Fault"/>
          </owl:intersectionOf>
        </owl:Class>
      </owl:equivalentClass>
    </owl:Class>
    <owl:Class rdf:ID="NonCriticalStatus">
      <owl:equivalentClass>
        <owl:Class>
          <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isStatusOf"/>
              <owl:someValuesFrom>
                <owl:Restriction>
                  <owl:onProperty rdf:resource="#hasObservation"/>
                  <owl:someValuesFrom>
                    <owl:Restriction>
                      <owl:onProperty rdf:resource="#hasSymptom"/>
                      <owl:someValuesFrom>
                        <owl:Restriction>
                          <owl:onProperty rdf:resource="#refersToFault"/>
                          <owl:allValuesFrom
rdf:resource="#NonCriticalFault"/>
                        </owl:Restriction>
                      </owl:someValuesFrom>
                    </owl:Restriction>
                  </owl:someValuesFrom>
                </owl:Restriction>
              </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isStatusOf"/>
              <owl:someValuesFrom>
                <owl:Restriction>
```

```
                        <owl:onProperty rdf:resource="#hasObservation"/>
                        <owl:someValuesFrom>
                           <owl:Restriction>
                              <owl:onProperty rdf:resource="#hasSymptom"/>
                              <owl:someValuesFrom>
                                 <owl:Restriction>
                                    <owl:onProperty rdf:resource="#refersToFault"/>
                                    <owl:someValuesFrom
rdf:resource="#NonCriticalFault"/>
                                 </owl:Restriction>
                              </owl:someValuesFrom>
                           </owl:Restriction>
                        </owl:someValuesFrom>
                     </owl:Restriction>
                  </owl:someValuesFrom>
               </owl:Restriction>
               <owl:Class rdf:about="#SystemStatus"/>
            </owl:intersectionOf>
         </owl:Class>
      </owl:equivalentClass>
   </owl:Class>
   <owl:Class rdf:ID="NonIncipientStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
   </owl:Class>
   <owl:Class rdf:ID="NonSelfPoweredVehicle">
      <rdfs:subClassOf rdf:resource="#Vehicle"/>
      <owl:disjointWith rdf:resource="#SelfPoweredVehicle"/>
   </owl:Class>
   <owl:Class rdf:ID="ObliqueCrossing">
      <rdfs:subClassOf rdf:resource="#Crossing"/>
      <owl:disjointWith rdf:resource="#Right-AngleCrossing"/>
      <owl:disjointWith rdf:resource="#DiamondCrossing"/>
   </owl:Class>
   <owl:Class rdf:ID="Observation">
      <owl:disjointWith rdf:resource="#ValuePartition"/>
      <owl:disjointWith rdf:resource="#LocationCoordinates"/>
      <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
      <owl:disjointWith rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#Observer"/>
      <owl:disjointWith rdf:resource="#Symptom"/>
```

```
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
  </owl:Class>
  <owl:Class rdf:ID="ObservationData">
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
  </owl:Class>
  <owl:Class rdf:ID="Observer">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#HumanObserver"/>
          <owl:Class rdf:about="#MeasurementSystem"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Symptom"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
  </owl:Class>
  <owl:Class rdf:ID="OperationalTrain">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFirstConsist"/>
            <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFirstConsist"/>
            <owl:allValuesFrom rdf:resource="#TrainConsist"/>
          </owl:Restriction>
          <owl:Class rdf:about="#RollingStockObject"/>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#Vehicle"/>
  </owl:Class>
  <owl:Class rdf:ID="OperationalTrainStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
```

```
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="order">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="&xsd;int"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="OverHeadLineObject">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
    <owl:disjointWith rdf:resource="#ElectricalAsset"/>
    <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
    <owl:disjointWith rdf:resource="#MechanicalAsset"/>
    <owl:disjointWith rdf:resource="#PointMachine"/>
    <owl:disjointWith rdf:resource="#PermanentWayObject"/>
    <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="ownsObservation">
    <rdfs:domain rdf:resource="#Observer"/>
    <rdfs:range rdf:resource="#Observation"/>
    <owl:inverseOf rdf:resource="#isObservationFrom"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="PassengerCar">
    <rdfs:subClassOf rdf:resource="#Railcar"/>
    <owl:disjointWith rdf:resource="#FreightCar"/>
</owl:Class>
<owl:Class rdf:ID="PassengerTrainConsist">
    <rdfs:subClassOf rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#FreightTrainConsist"/>
</owl:Class>
<owl:Class rdf:ID="PermanentWayObject">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
    <owl:disjointWith rdf:resource="#ElectricalAsset"/>
    <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
    <owl:disjointWith rdf:resource="#MechanicalAsset"/>
    <owl:disjointWith rdf:resource="#PointMachine"/>
    <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
    <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
</owl:Class>
<owl:Class rdf:ID="Platform">
    <rdfs:subClassOf rdf:resource="#TrackNetworkNode"/>
    <owl:disjointWith rdf:resource="#BufferStop"/>
    <owl:disjointWith rdf:resource="#SandDrag"/>
    <owl:disjointWith rdf:resource="#Signal"/>
    <owl:disjointWith rdf:resource="#TerraIncognita"/>
</owl:Class>
<owl:Class rdf:ID="PneumaticDoorSystem">
    <rdfs:subClassOf rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#ElectricDoorSystem"/>
    <owl:disjointWith rdf:resource="#ManualDoorSystem"/>
</owl:Class>
<owl:Class rdf:ID="PointMachine">
    <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
    <owl:disjointWith rdf:resource="#ElectricalAsset"/>
    <owl:disjointWith rdf:resource="#MechanicalAsset"/>
    <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
    <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
    <owl:disjointWith rdf:resource="#PermanentWayObject"/>
    <owl:disjointWith rdf:resource="#SignallingSystemsObject"/>
</owl:Class>
<owl:Class rdf:ID="PointMachineStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
```

```
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="PointsStatus">
        <rdfs:subClassOf rdf:resource="#SystemStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="PowerSystem">
        <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
        <owl:disjointWith rdf:resource="#DoorSystem"/>
        <owl:disjointWith rdf:resource="#Window"/>
```

```
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
</owl:Class>
<owl:Class rdf:ID="PowerSystemComponent">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
</owl:Class>
    <owl:Class rdf:ID="PowerSystemStatus">
        <rdfs:subClassOf rdf:resource="#SystemStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#RollingStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#DriveStatus"/>
        <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#TrackStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#PointsStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="PrimarySuspension">
```

```
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
  </owl:Class>
  <owl:Class rdf:ID="PrimarySuspensionStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
  </owl:Class>
  <owl:Class rdf:ID="PropulsionSystem">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
```

```
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
  </owl:Class>
  <owl:Class rdf:ID="PropulsionSystemComponent">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
  </owl:Class>
  <owl:Class rdf:ID="PropulsionSystemStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
  </owl:Class>
  <owl:Class rdf:ID="Railcar">
    <rdfs:subClassOf rdf:resource="#NonSelfPoweredVehicle"/>
  </owl:Class>
  <owl:Class rdf:ID="RailwayLocationCoordinates">
    <rdfs:subClassOf rdf:resource="#Location"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
    <owl:disjointWith rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#Symptom"/>
```

```
    </owl:Class>
    <owl:ObjectProperty rdf:ID="refersToFault">
      <rdfs:domain rdf:resource="#Symptom"/>
      <rdfs:range rdf:resource="#Fault"/>
      <owl:inverseOf rdf:resource="#causedBySymptom"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="refersToObservation">
      <rdfs:domain rdf:resource="#Symptom"/>
      <rdfs:range rdf:resource="#Observation"/>
      <owl:inverseOf rdf:resource="#hasSymptom"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="refersToSystem">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:domain rdf:resource="#Observation"/>
      <rdfs:range rdf:resource="#TargetSystem"/>
      <owl:inverseOf rdf:resource="#hasObservation"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="RegionalBoundary">
      <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
      <owl:disjointWith rdf:resource="#Depot"/>
      <owl:disjointWith rdf:resource="#FreightTerminal"/>
      <owl:disjointWith rdf:resource="#Junction"/>
      <owl:disjointWith rdf:resource="#Siding"/>
      <owl:disjointWith rdf:resource="#Station"/>
      <owl:disjointWith rdf:resource="#Tunnel"/>
    </owl:Class>
    <Reverse rdf:ID="rev"/>
    <owl:Class rdf:ID="Reverse">
      <rdfs:subClassOf rdf:resource="#Direction"/>
      <owl:disjointWith rdf:resource="#Forward"/>
    </owl:Class>
    <owl:Class rdf:ID="RideQualityStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="Right">
      <rdfs:subClassOf rdf:resource="#Side"/>
      <owl:disjointWith rdf:resource="#Left"/>
    </owl:Class>
    <Right rdf:ID="right"/>
    <owl:Class rdf:ID="Right-AngleCrossing">
```

```
      <rdfs:subClassOf rdf:resource="#Crossing"/>
      <owl:disjointWith rdf:resource="#ObliqueCrossing"/>
      <owl:disjointWith rdf:resource="#DiamondCrossing"/>
   </owl:Class>
   <owl:Class rdf:ID="RollingStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
   </owl:Class>
   <owl:Class rdf:ID="RollingStockObject">
      <rdfs:subClassOf rdf:resource="#TargetSystem"/>
      <owl:disjointWith rdf:resource="#NetworkTopologyElement"/>
   </owl:Class>
   <owl:Class rdf:ID="Route">
      <owl:disjointWith rdf:resource="#Symptom"/>
      <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
      <owl:disjointWith rdf:resource="#ValuePartition"/>
      <owl:disjointWith rdf:resource="#Observation"/>
      <owl:disjointWith rdf:resource="#TargetSystem"/>
      <owl:disjointWith rdf:resource="#ObservationData"/>
      <owl:disjointWith rdf:resource="#Fault"/>
      <owl:disjointWith rdf:resource="#Observer"/>
      <owl:disjointWith rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
      <owl:disjointWith rdf:resource="#LocationCoordinates"/>
   </owl:Class>
   <owl:Class rdf:ID="RouteNetworkEdge">
      <rdfs:subClassOf rdf:resource="&owl;Thing"/>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#endsAt"/>
            <owl:someValuesFrom rdf:resource="#LineNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#endsAt"/>
            <owl:allValuesFrom rdf:resource="#LineNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="#startsAt"/>
            <owl:allValuesFrom rdf:resource="#LineNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#startsAt"/>
            <owl:someValuesFrom rdf:resource="#LineNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <owl:disjointWith rdf:resource="#ObservationData"/>
      <owl:disjointWith rdf:resource="#Symptom"/>
      <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
      <owl:disjointWith rdf:resource="#Observation"/>
      <owl:disjointWith rdf:resource="#Fault"/>
      <owl:disjointWith rdf:resource="#Observer"/>
      <owl:disjointWith rdf:resource="#ValuePartition"/>
      <owl:disjointWith rdf:resource="#Route"/>
      <owl:disjointWith rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#LocationCoordinates"/>
   </owl:Class>
   <owl:Class rdf:ID="RouteSegment">
      <owl:equivalentClass>
         <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#isMappedToLineOfRoute"/>
                  <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
               </owl:Restriction>
               <owl:Restriction>
                  <owl:onProperty rdf:resource="#isMappedToLineOfRoute"/>
                  <owl:allValuesFrom rdf:resource="#Track"/>
               </owl:Restriction>
               <owl:Class rdf:about="#RouteNetworkEdge"/>
            </owl:intersectionOf>
         </owl:Class>
      </owl:equivalentClass>
   </owl:Class>
   <owl:Class rdf:ID="SandDrag">
      <rdfs:subClassOf rdf:resource="#TrackNetworkNode"/>
      <owl:disjointWith rdf:resource="#BufferStop"/>
      <owl:disjointWith rdf:resource="#Platform"/>
      <owl:disjointWith rdf:resource="#Signal"/>
      <owl:disjointWith rdf:resource="#TerraIncognita"/>
   </owl:Class>
   <owl:Class rdf:ID="SecondarySuspension">
      <rdfs:subClassOf rdf:resource="#BogieComponent"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
      <owl:disjointWith rdf:resource="#Transmission"/>
      <owl:disjointWith rdf:resource="#Bearing"/>
      <owl:disjointWith rdf:resource="#BogieLinkage"/>
      <owl:disjointWith rdf:resource="#WheelSet"/>
      <owl:disjointWith rdf:resource="#BogieFrame"/>
      <owl:disjointWith rdf:resource="#Axle"/>
      <owl:disjointWith rdf:resource="#MotorInterface"/>
      <owl:disjointWith rdf:resource="#BearingBox"/>
      <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
      <owl:disjointWith rdf:resource="#BearingAssembly"/>
      <owl:disjointWith rdf:resource="#Wheel"/>
      <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
      <owl:disjointWith rdf:resource="#GearBoxInterface"/>
      <owl:disjointWith rdf:resource="#CoolingSystem"/>
      <owl:disjointWith rdf:resource="#PrimarySuspension"/>
      <owl:disjointWith rdf:resource="#WheelSetSystem"/>
   </owl:Class>
   <owl:Class rdf:ID="SecondarySuspensionStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#TrackStatus"/>
```

```
        <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
        <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
        <owl:disjointWith rdf:resource="#WheelStatus"/>
        <owl:disjointWith rdf:resource="#CarBodyStatus"/>
        <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
        <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
        <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
        <owl:disjointWith rdf:resource="#JunctionStatus"/>
        <owl:disjointWith rdf:resource="#RideQualityStatus"/>
        <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
        <owl:disjointWith rdf:resource="#WheelSetStatus"/>
        <owl:disjointWith rdf:resource="#CarriageStatus"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
        <owl:disjointWith rdf:resource="#VehicleStatus"/>
        <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
        <owl:disjointWith rdf:resource="#AccelerationStatus"/>
        <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
        <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
        <owl:disjointWith rdf:resource="#TransmissionStatus"/>
        <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
        <owl:disjointWith rdf:resource="#PointMachineStatus"/>
        <owl:disjointWith rdf:resource="#BogieStatus"/>
        <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
        <owl:disjointWith rdf:resource="#AxleStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="SelfPoweredRailcar">
        <rdfs:subClassOf rdf:resource="#SelfPoweredVehicle"/>
        <owl:disjointWith rdf:resource="#Locomotive"/>
    </owl:Class>
    <owl:Class rdf:ID="SelfPoweredVehicle">
        <rdfs:subClassOf rdf:resource="#Vehicle"/>
        <owl:disjointWith rdf:resource="#NonSelfPoweredVehicle"/>
    </owl:Class>
    <owl:Class rdf:ID="Side">
        <owl:equivalentClass>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <owl:Class rdf:about="#Left"/>
                    <owl:Class rdf:about="#Right"/>
                </owl:unionOf>
            </owl:Class>
        </owl:equivalentClass>
        <rdfs:subClassOf rdf:resource="#ValuePartition"/>
    </owl:Class>
    <owl:Class rdf:ID="Siding">
        <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
        <owl:disjointWith rdf:resource="#Depot"/>
        <owl:disjointWith rdf:resource="#FreightTerminal"/>
        <owl:disjointWith rdf:resource="#Junction"/>
        <owl:disjointWith rdf:resource="#RegionalBoundary"/>
        <owl:disjointWith rdf:resource="#Station"/>
        <owl:disjointWith rdf:resource="#Tunnel"/>
    </owl:Class>
    <owl:Class rdf:ID="Signal">
        <rdfs:subClassOf rdf:resource="#TrackNetworkNode"/>
        <owl:disjointWith rdf:resource="#BufferStop"/>
        <owl:disjointWith rdf:resource="#Platform"/>
        <owl:disjointWith rdf:resource="#SandDrag"/>
        <owl:disjointWith rdf:resource="#TerraIncognita"/>
    </owl:Class>
    <owl:Class rdf:ID="SignallingSystemsObject">
        <rdfs:subClassOf rdf:resource="#NetworkTopologyAsset"/>
        <owl:disjointWith rdf:resource="#ElectricalAsset"/>
        <owl:disjointWith rdf:resource="#ElectromechanicalAsset"/>
        <owl:disjointWith rdf:resource="#MechanicalAsset"/>
        <owl:disjointWith rdf:resource="#PointMachine"/>
        <owl:disjointWith rdf:resource="#OverHeadLineObject"/>
        <owl:disjointWith rdf:resource="#PermanentWayObject"/>
    </owl:Class>
```

```
<owl:Class rdf:ID="SpeedometerSystem">
   <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
   <owl:disjointWith rdf:resource="#PropulsionSystem"/>
   <owl:disjointWith rdf:resource="#PowerSystem"/>
   <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
   <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
   <owl:disjointWith rdf:resource="#DoorSystem"/>
   <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
   <owl:disjointWith rdf:resource="#Window"/>
   <owl:disjointWith rdf:resource="#BrakingSystem"/>
   <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
   <owl:disjointWith rdf:resource="#Vehicle"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
   <owl:disjointWith rdf:resource="#TrainConsist"/>
   <owl:disjointWith rdf:resource="#Bogie"/>
   <owl:disjointWith rdf:resource="#BogieComponent"/>
   <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
   <owl:disjointWith rdf:resource="#CarBody"/>
</owl:Class>
<owl:Class rdf:ID="SpeedometerSystemComponent">
   <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
   <owl:disjointWith rdf:resource="#Window"/>
   <owl:disjointWith rdf:resource="#PowerSystem"/>
   <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
   <owl:disjointWith rdf:resource="#BrakingSystem"/>
   <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
   <owl:disjointWith rdf:resource="#TrainConsist"/>
   <owl:disjointWith rdf:resource="#PropulsionSystem"/>
   <owl:disjointWith rdf:resource="#CarBody"/>
   <owl:disjointWith rdf:resource="#Bogie"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
   <owl:disjointWith rdf:resource="#BogieComponent"/>
   <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
   <owl:disjointWith rdf:resource="#DoorSystem"/>
   <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
   <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
   <owl:disjointWith rdf:resource="#Vehicle"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
</owl:Class>
<owl:Class rdf:ID="SpeedometerSystemStatus">
   <rdfs:subClassOf rdf:resource="#SystemStatus"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
   <owl:disjointWith rdf:resource="#AccelerationStatus"/>
   <owl:disjointWith rdf:resource="#TrackStatus"/>
   <owl:disjointWith rdf:resource="#RollingStatus"/>
   <owl:disjointWith rdf:resource="#VehicleStatus"/>
   <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#AxleStatus"/>
   <owl:disjointWith rdf:resource="#PointsStatus"/>
   <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
   <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
   <owl:disjointWith rdf:resource="#RideQualityStatus"/>
   <owl:disjointWith rdf:resource="#CarriageStatus"/>
   <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
   <owl:disjointWith rdf:resource="#TransmissionStatus"/>
   <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
   <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
   <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
   <owl:disjointWith rdf:resource="#DriveStatus"/>
   <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#BogieStatus"/>
   <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
   <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
   <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
   <owl:disjointWith rdf:resource="#WheelStatus"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
   <owl:disjointWith rdf:resource="#CarBodyStatus"/>
   <owl:disjointWith rdf:resource="#WheelSetStatus"/>
   <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
   <owl:disjointWith rdf:resource="#JunctionStatus"/>
```

```
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
</owl:Class>
<owl:Class rdf:ID="StandardFreightTrainConsist">
    <rdfs:subClassOf rdf:resource="#FreightTrainConsist"/>
    <owl:disjointWith rdf:resource="#Class4FreightTrainConsist"/>
</owl:Class>
<owl:Class rdf:ID="StandardPassengerTrainConsist">
    <rdfs:subClassOf rdf:resource="#PassengerTrainConsist"/>
    <owl:disjointWith rdf:resource="#HighSpeedPassengerTrainConsist"/>
</owl:Class>
<owl:Class rdf:ID="StandardTurnout">
    <rdfs:subClassOf rdf:resource="#Switch"/>
    <owl:disjointWith rdf:resource="#SymmetricalTurnout"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="startsAt">
    <rdfs:domain>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#NetworkTopologyEdge"/>
                <owl:Class rdf:about="#RouteNetworkEdge"/>
            </owl:unionOf>
        </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#NetworkTopologyNode"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Station">
    <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
    <owl:disjointWith rdf:resource="#Depot"/>
    <owl:disjointWith rdf:resource="#FreightTerminal"/>
    <owl:disjointWith rdf:resource="#Junction"/>
    <owl:disjointWith rdf:resource="#RegionalBoundary"/>
    <owl:disjointWith rdf:resource="#Siding"/>
    <owl:disjointWith rdf:resource="#Tunnel"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="statusReliesOn">
    <rdfs:domain rdf:resource="#SystemStatus"/>
    <rdfs:range rdf:resource="#SystemStatus"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Steady">
    <rdfs:subClassOf rdf:resource="#Trend"/>
    <owl:disjointWith rdf:resource="#Increasing"/>
    <owl:disjointWith rdf:resource="#Decreasing"/>
</owl:Class>
<Steady rdf:ID="steady"/>
<owl:Class rdf:ID="Switch">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#StandardTurnout"/>
                        <owl:Class rdf:about="#SymmetricalTurnout"/>
                    </owl:unionOf>
                </owl:Class>
                <owl:Class rdf:about="#TrackNetworkNode"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="SymmetricalTurnout">
    <rdfs:subClassOf rdf:resource="#Switch"/>
    <owl:disjointWith rdf:resource="#StandardTurnout"/>
</owl:Class>
<owl:Class rdf:ID="Symptom">
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#refersToFault"/>
            <owl:someValuesFrom rdf:resource="#Fault"/>
        </owl:Restriction>
    </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#refersToFault"/>
        <owl:allValuesFrom rdf:resource="#Fault"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
    <owl:disjointWith rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="systemReliesOn">
    <rdfs:domain rdf:resource="#TargetSystem"/>
    <rdfs:range rdf:resource="#TargetSystem"/>
  </owl:ObjectProperty>
  <owl:Class rdf:ID="SystemStatus">
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="TargetSystem">
    <owl:equivalentClass>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#NetworkTopologyElement"/>
          <owl:Class rdf:about="#RollingStockObject"/>
        </owl:unionOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasStatus"/>
        <owl:allValuesFrom rdf:resource="#SystemStatus"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="&owl;Thing"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasStatus"/>
        <owl:someValuesFrom rdf:resource="#SystemStatus"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#ValuePartition"/>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Fault"/>
    <owl:disjointWith rdf:resource="#ObservationData"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#Symptom"/>
  </owl:Class>
  <owl:Class rdf:ID="TerraIncognita">
    <rdfs:subClassOf rdf:resource="#TrackNetworkNode"/>
    <owl:disjointWith rdf:resource="#BufferStop"/>
    <owl:disjointWith rdf:resource="#Platform"/>
    <owl:disjointWith rdf:resource="#SandDrag"/>
    <owl:disjointWith rdf:resource="#Signal"/>
  </owl:Class>
```

```
<owl:DatatypeProperty rdf:ID="timeStamp">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#Observation"/>
   <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="TorqueTransmissionDeviceConnector">
   <rdfs:subClassOf rdf:resource="#BogieComponent"/>
   <owl:disjointWith rdf:resource="#Bearing"/>
   <owl:disjointWith rdf:resource="#MotorInterface"/>
   <owl:disjointWith rdf:resource="#GearBoxInterface"/>
   <owl:disjointWith rdf:resource="#BogieLinkage"/>
   <owl:disjointWith rdf:resource="#SecondarySuspension"/>
   <owl:disjointWith rdf:resource="#BearingBox"/>
   <owl:disjointWith rdf:resource="#BogieFrame"/>
   <owl:disjointWith rdf:resource="#PrimarySuspension"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
   <owl:disjointWith rdf:resource="#Axle"/>
   <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
   <owl:disjointWith rdf:resource="#BearingAssembly"/>
   <owl:disjointWith rdf:resource="#CoolingSystem"/>
   <owl:disjointWith rdf:resource="#Wheel"/>
   <owl:disjointWith rdf:resource="#WheelSet"/>
   <owl:disjointWith rdf:resource="#WheelSetSystem"/>
   <owl:disjointWith rdf:resource="#Transmission"/>
</owl:Class>
<owl:Class rdf:ID="TorqueTransmissionDeviceConnectorStatus">
   <rdfs:subClassOf rdf:resource="#SystemStatus"/>
   <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
   <owl:disjointWith rdf:resource="#WheelSetStatus"/>
   <owl:disjointWith rdf:resource="#WheelStatus"/>
   <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
   <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
   <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
   <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
   <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
   <owl:disjointWith rdf:resource="#TrackStatus"/>
   <owl:disjointWith rdf:resource="#RollingStatus"/>
   <owl:disjointWith rdf:resource="#BogieStatus"/>
   <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#CarBodyStatus"/>
   <owl:disjointWith rdf:resource="#TransmissionStatus"/>
   <owl:disjointWith rdf:resource="#AxleStatus"/>
   <owl:disjointWith rdf:resource="#RideQualityStatus"/>
   <owl:disjointWith rdf:resource="#AccelerationStatus"/>
   <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
   <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
   <owl:disjointWith rdf:resource="#JunctionStatus"/>
   <owl:disjointWith rdf:resource="#PointMachineStatus"/>
   <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
   <owl:disjointWith rdf:resource="#PointsStatus"/>
   <owl:disjointWith rdf:resource="#VehicleStatus"/>
   <owl:disjointWith rdf:resource="#DriveStatus"/>
   <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
   <owl:disjointWith rdf:resource="#CarriageStatus"/>
   <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
</owl:Class>
<owl:Class rdf:ID="Track">
   <rdfs:subClassOf rdf:resource="#TrackNetworkEdge"/>
</owl:Class>
<owl:Class rdf:ID="TrackNetworkEdge">
   <rdfs:subClassOf rdf:resource="#NetworkTopologyEdge"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#startsAt"/>
         <owl:someValuesFrom rdf:resource="#TrackNetworkNode"/>
      </owl:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <owl:Restriction>
```

```
            <owl:onProperty rdf:resource="#startsAt"/>
            <owl:allValuesFrom rdf:resource="#TrackNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#endsAt"/>
            <owl:someValuesFrom rdf:resource="#TrackNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#endsAt"/>
            <owl:allValuesFrom rdf:resource="#TrackNetworkNode"/>
         </owl:Restriction>
      </rdfs:subClassOf>
      <owl:disjointWith rdf:resource="#LineNetworkEdge"/>
   </owl:Class>
   <owl:Class rdf:ID="TrackNetworkNode">
      <owl:equivalentClass>
         <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
               <owl:Class>
                  <owl:unionOf rdf:parseType="Collection">
                     <owl:Class rdf:about="#BufferStop"/>
                     <owl:Class rdf:about="#Crossing"/>
                     <owl:Class rdf:about="#Platform"/>
                     <owl:Class rdf:about="#SandDrag"/>
                     <owl:Class rdf:about="#Signal"/>
                     <owl:Class rdf:about="#Switch"/>
                     <owl:Class rdf:about="#TerraIncognita"/>
                  </owl:unionOf>
               </owl:Class>
               <owl:Class rdf:about="#NetworkTopologyNode"/>
            </owl:intersectionOf>
         </owl:Class>
      </owl:equivalentClass>
      <owl:disjointWith rdf:resource="#LineNetworkNode"/>
   </owl:Class>
   <owl:Class rdf:ID="TrackStatus">
      <rdfs:subClassOf rdf:resource="#SystemStatus"/>
      <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
      <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
      <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
      <owl:disjointWith rdf:resource="#VehicleStatus"/>
      <owl:disjointWith rdf:resource="#RollingStatus"/>
      <owl:disjointWith rdf:resource="#AccelerationStatus"/>
      <owl:disjointWith rdf:resource="#DriveStatus"/>
      <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
      <owl:disjointWith rdf:resource="#BogieStatus"/>
      <owl:disjointWith rdf:resource="#TransmissionStatus"/>
      <owl:disjointWith rdf:resource="#RideQualityStatus"/>
      <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
      <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
      <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
      <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
      <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
      <owl:disjointWith rdf:resource="#AxleStatus"/>
      <owl:disjointWith rdf:resource="#PointMachineStatus"/>
      <owl:disjointWith rdf:resource="#WheelSetStatus"/>
      <owl:disjointWith rdf:resource="#CarBodyStatus"/>
      <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
      <owl:disjointWith rdf:resource="#WheelStatus"/>
      <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
      <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
      <owl:disjointWith rdf:resource="#JunctionStatus"/>
      <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
      <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
      <owl:disjointWith rdf:resource="#PointsStatus"/>
      <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
      <owl:disjointWith rdf:resource="#CarriageStatus"/>
```

```
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
</owl:Class>
<owl:Class rdf:ID="TrailerBogie">
    <rdfs:subClassOf rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#MotorBogie"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="train_ID">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#OperationalTrain"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="TrainBorneMeasurementSystem">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isLocatedAt"/>
                    <owl:allValuesFrom rdf:resource="#RollingStockObject"/>
                </owl:Restriction>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#isLocatedAt"/>
                    <owl:someValuesFrom rdf:resource="#RollingStockObject"/>
                </owl:Restriction>
                <owl:Class rdf:about="#MeasurementSystem"/>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="TrainConsist">
    <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFirstVehicle"/>
            <owl:allValuesFrom rdf:resource="#Vehicle"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#hasFirstVehicle"/>
            <owl:cardinality rdf:datatype="&xsd;int">1</owl:cardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#Vehicle"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
</owl:Class>
<owl:Class rdf:ID="TrainConsistStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
```

```
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
</owl:Class>
<owl:DatatypeProperty rdf:ID="trainLength">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#TrainConsist"/>
    <rdfs:range rdf:resource="&xsd;float"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="TrainService"/>
<owl:Class rdf:ID="Transmission">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
  </owl:Class>
  <owl:Class rdf:ID="TransmissionStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
```

```
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
</owl:Class>
<owl:Class rdf:ID="Trend">
    <owl:equivalentClass>
        <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Decreasing"/>
                <owl:Class rdf:about="#Increasing"/>
                <owl:Class rdf:about="#Steady"/>
            </owl:unionOf>
        </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#ValuePartition"/>
</owl:Class>
<owl:Class rdf:ID="Tunnel">
    <rdfs:subClassOf rdf:resource="#LineNetworkNode"/>
    <owl:disjointWith rdf:resource="#Depot"/>
    <owl:disjointWith rdf:resource="#FreightTerminal"/>
    <owl:disjointWith rdf:resource="#Junction"/>
    <owl:disjointWith rdf:resource="#RegionalBoundary"/>
    <owl:disjointWith rdf:resource="#Siding"/>
    <owl:disjointWith rdf:resource="#Station"/>
</owl:Class>
<owl:Class rdf:ID="ValuePartition">
    <owl:disjointWith rdf:resource="#TargetSystem"/>
    <owl:disjointWith rdf:resource="#Observer"/>
    <owl:disjointWith rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#Route"/>
    <owl:disjointWith rdf:resource="#Observation"/>
    <owl:disjointWith rdf:resource="#RouteNetworkEdge"/>
    <owl:disjointWith rdf:resource="#ObservationData"/>
    <owl:disjointWith rdf:resource="#LocationCoordinates"/>
    <owl:disjointWith rdf:resource="#RailwayLocationCoordinates"/>
    <owl:disjointWith rdf:resource="#Fault"/>
</owl:Class>
<owl:Class rdf:ID="Vehicle">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#RollingStockObject"/>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <owl:Class rdf:about="#NonSelfPoweredVehicle"/>
                        <owl:Class rdf:about="#SelfPoweredVehicle"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
    <owl:disjointWith rdf:resource="#OperationalTrain"/>
    <owl:disjointWith rdf:resource="#PropulsionSystem"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
    <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
    <owl:disjointWith rdf:resource="#TrainConsist"/>
    <owl:disjointWith rdf:resource="#DoorSystem"/>
    <owl:disjointWith rdf:resource="#Bogie"/>
    <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
    <owl:disjointWith rdf:resource="#Window"/>
    <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
```

```
    <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
    <owl:disjointWith rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#PowerSystem"/>
    <owl:disjointWith rdf:resource="#CarBody"/>
    <owl:disjointWith rdf:resource="#BrakingSystem"/>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="vehicle_ID">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="&xsd;string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="vehicleLength">
    <rdf:type rdf:resource="&owl;FunctionalProperty"/>
    <rdfs:domain rdf:resource="#Vehicle"/>
    <rdfs:range rdf:resource="&xsd;float"/>
  </owl:DatatypeProperty>
  <owl:Class rdf:ID="VehicleStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#WheelStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
  </owl:Class>
  <owl:Class rdf:ID="Wheel">
    <rdfs:subClassOf rdf:resource="#BogieComponent"/>
    <owl:disjointWith rdf:resource="#GearBoxInterface"/>
    <owl:disjointWith rdf:resource="#Axle"/>
    <owl:disjointWith rdf:resource="#SecondarySuspension"/>
    <owl:disjointWith rdf:resource="#WheelSetSystem"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
    <owl:disjointWith rdf:resource="#BearingAssembly"/>
    <owl:disjointWith rdf:resource="#Bearing"/>
    <owl:disjointWith rdf:resource="#MotorInterface"/>
    <owl:disjointWith rdf:resource="#PrimarySuspension"/>
    <owl:disjointWith rdf:resource="#WheelSet"/>
    <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
    <owl:disjointWith rdf:resource="#BogieFrame"/>
    <owl:disjointWith rdf:resource="#Transmission"/>
    <owl:disjointWith rdf:resource="#BearingBox"/>
    <owl:disjointWith rdf:resource="#CoolingSystem"/>
    <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
  </owl:Class>
```

```
<owl:Class rdf:ID="WheelSet">
   <rdfs:subClassOf rdf:resource="#BogieComponent"/>
   <owl:disjointWith rdf:resource="#SecondarySuspension"/>
   <owl:disjointWith rdf:resource="#Transmission"/>
   <owl:disjointWith rdf:resource="#MotorInterface"/>
   <owl:disjointWith rdf:resource="#CoolingSystem"/>
   <owl:disjointWith rdf:resource="#BogieFrame"/>
   <owl:disjointWith rdf:resource="#BearingBox"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
   <owl:disjointWith rdf:resource="#GearBoxInterface"/>
   <owl:disjointWith rdf:resource="#Bearing"/>
   <owl:disjointWith rdf:resource="#WheelSetSystem"/>
   <owl:disjointWith rdf:resource="#Axle"/>
   <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
   <owl:disjointWith rdf:resource="#BogieLinkage"/>
   <owl:disjointWith rdf:resource="#BearingAssembly"/>
   <owl:disjointWith rdf:resource="#Wheel"/>
   <owl:disjointWith rdf:resource="#PrimarySuspension"/>
   <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
</owl:Class>
<owl:Class rdf:ID="WheelSetStatus">
   <rdfs:subClassOf rdf:resource="#SystemStatus"/>
   <owl:disjointWith rdf:resource="#RideQualityStatus"/>
   <owl:disjointWith rdf:resource="#TrackStatus"/>
   <owl:disjointWith rdf:resource="#CarBodyStatus"/>
   <owl:disjointWith rdf:resource="#TransmissionStatus"/>
   <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
   <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#WheelStatus"/>
   <owl:disjointWith rdf:resource="#RollingStatus"/>
   <owl:disjointWith rdf:resource="#PointMachineStatus"/>
   <owl:disjointWith rdf:resource="#AxleStatus"/>
   <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
   <owl:disjointWith rdf:resource="#CarriageStatus"/>
   <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
   <owl:disjointWith rdf:resource="#PointsStatus"/>
   <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
   <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
   <owl:disjointWith rdf:resource="#DriveStatus"/>
   <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
   <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
   <owl:disjointWith rdf:resource="#AccelerationStatus"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
   <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
   <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
   <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
   <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
   <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
   <owl:disjointWith rdf:resource="#JunctionStatus"/>
   <owl:disjointWith rdf:resource="#VehicleStatus"/>
   <owl:disjointWith rdf:resource="#BogieStatus"/>
   <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
   <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
</owl:Class>
<owl:Class rdf:ID="WheelSetSystem">
   <rdfs:subClassOf rdf:resource="#BogieComponent"/>
   <owl:disjointWith rdf:resource="#PrimarySuspension"/>
   <owl:disjointWith rdf:resource="#CoolingSystem"/>
   <owl:disjointWith rdf:resource="#BearingBox"/>
   <owl:disjointWith rdf:resource="#BrakeDiskInterface"/>
   <owl:disjointWith rdf:resource="#MotorInterface"/>
   <owl:disjointWith rdf:resource="#TorqueTransmissionDeviceConnector"/>
   <owl:disjointWith rdf:resource="#BogieFrame"/>
   <owl:disjointWith rdf:resource="#BearingAssembly"/>
   <owl:disjointWith rdf:resource="#WheelSet"/>
   <owl:disjointWith rdf:resource="#AuxiliarySystem"/>
   <owl:disjointWith rdf:resource="#GearBoxInterface"/>
   <owl:disjointWith rdf:resource="#Axle"/>
   <owl:disjointWith rdf:resource="#Transmission"/>
   <owl:disjointWith rdf:resource="#Bearing"/>
   <owl:disjointWith rdf:resource="#SecondarySuspension"/>
```

```
    <owl:disjointWith rdf:resource="#Wheel"/>
    <owl:disjointWith rdf:resource="#BogieLinkage"/>
</owl:Class>
<owl:Class rdf:ID="WheelStatus">
    <rdfs:subClassOf rdf:resource="#SystemStatus"/>
    <owl:disjointWith rdf:resource="#AuxiliarySystemStatus"/>
    <owl:disjointWith rdf:resource="#FireDetectionSystemStatus"/>
    <owl:disjointWith rdf:resource="#PropulsionSystemStatus"/>
    <owl:disjointWith rdf:resource="#AccelerationStatus"/>
    <owl:disjointWith rdf:resource="#CoolingSystemStatus"/>
    <owl:disjointWith rdf:resource="#PointMachineStatus"/>
    <owl:disjointWith rdf:resource="#DoorSystemStatus"/>
    <owl:disjointWith rdf:resource="#SpeedometerSystemStatus"/>
    <owl:disjointWith rdf:resource="#OperationalTrainStatus"/>
    <owl:disjointWith rdf:resource="#NonIncipientStatus"/>
    <owl:disjointWith rdf:resource="#TrainConsistStatus"/>
    <owl:disjointWith rdf:resource="#VehicleStatus"/>
    <owl:disjointWith rdf:resource="#AxleBearingStatus"/>
    <owl:disjointWith rdf:resource="#CarBodyStatus"/>
    <owl:disjointWith rdf:resource="#TrackStatus"/>
    <owl:disjointWith rdf:resource="#PointsStatus"/>
    <owl:disjointWith rdf:resource="#AxleStatus"/>
    <owl:disjointWith rdf:resource="#PrimarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#TransmissionStatus"/>
    <owl:disjointWith rdf:resource="#PowerSystemStatus"/>
    <owl:disjointWith rdf:resource="#CarriageStatus"/>
    <owl:disjointWith rdf:resource="#BrakingSystemStatus"/>
    <owl:disjointWith rdf:resource="#WheelSetStatus"/>
    <owl:disjointWith rdf:resource="#DriveStatus"/>
    <owl:disjointWith rdf:resource="#BogieStatus"/>
    <owl:disjointWith rdf:resource="#SecondarySuspensionStatus"/>
    <owl:disjointWith rdf:resource="#JunctionStatus"/>
    <owl:disjointWith rdf:resource="#RideQualityStatus"/>
    <owl:disjointWith rdf:resource="#RollingStatus"/>
    <owl:disjointWith rdf:resource="#LineOfRouteStatus"/>
    <owl:disjointWith
rdf:resource="#TorqueTransmissionDeviceConnectorStatus"/>
    </owl:Class>
    <owl:Class rdf:ID="Window">
        <rdfs:subClassOf rdf:resource="#RollingStockObject"/>
        <owl:disjointWith rdf:resource="#Bogie"/>
        <owl:disjointWith rdf:resource="#BrakingSystemComponent"/>
        <owl:disjointWith rdf:resource="#PropulsionSystem"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystem"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystemComponent"/>
        <owl:disjointWith rdf:resource="#BrakingSystem"/>
        <owl:disjointWith rdf:resource="#PropulsionSystemComponent"/>
        <owl:disjointWith rdf:resource="#Vehicle"/>
        <owl:disjointWith rdf:resource="#SpeedometerSystemComponent"/>
        <owl:disjointWith rdf:resource="#PowerSystemComponent"/>
        <owl:disjointWith rdf:resource="#DoorSystemComponent"/>
        <owl:disjointWith rdf:resource="#TrainConsist"/>
        <owl:disjointWith rdf:resource="#CarBody"/>
        <owl:disjointWith rdf:resource="#BogieComponent"/>
        <owl:disjointWith rdf:resource="#DoorSystem"/>
        <owl:disjointWith rdf:resource="#FireDetectionSystem"/>
        <owl:disjointWith rdf:resource="#PowerSystem"/>
    </owl:Class>
    <owl:DatatypeProperty rdf:ID="Yards">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:domain rdf:resource="#RailwayLocationCoordinates"/>
        <rdfs:range rdf:resource="&xsd;string"/>
    </owl:DatatypeProperty>
</rdf:RDF>
```

A

Amdahl, G., Brooks, F., & Blaauw, G. A., “Architecture of the IBM System/360” Reprinted from IBM Journal of Research and Development, vol. 8, no. 2, 1964; ©1964, 2000 IBM j. res. develop. vol. 44 no. 1/2 January/March 2000.

Anderson, R., Hirsch, R., Trompet, M., & Adeney, W., “Developing Benchmarking Methodologies For Railway Infrastructure Management Companies,” Railway Technology Strategy Centre, Centre for Transport Studies, Imperial College London, UK

Antoniou, G., Van Harmelen, F., “A Semantic Web Primer,” MIT Press 2004, ISBN-978-0262012102.

B

Ba, S., Lang, K. R. & Whinston, A. B., “Enterprise decision support using intranet technology.” *Decision Support Systems.* 20, 2 (June 1997), 99-134.

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. F., (Eds.). (2003). The description logic handbook: Theory, implementation and applications. Cambridge University Press.

Baader, F., Horrocks, I. & Sattler, U., “Description Logics.” In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation, chapter 3, pages 135-180. Elsevier, 2008.

Baclawski, K., Kokar, M. M., Waldinger, R. J., & Kogut, P. A., Consistency Checking of Semantic Web Ontologies. In Proceedings of the First International Semantic Web Conference on The Semantic Web (ISWC '02), Ian Horrocks and James A. Hendler (Eds.). Springer-Verlag, London, UK, UK, 454-459.

Bechofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., Stein, L. A., "OWL Web Ontology Language," W3C Recommendation 10 February 2004 Referencehttp://www.w3.org/TR/2004/REC-owl-ref-20040210/

Berners-Lee, T. Hendler, J. & Lassila, O., "The Semantic Web," Scientific Am., May 2001, pp. 34–43.

Bizer, C. & Cyganiak, R., "D2R-Server - Publishing Relational Databases on the Web as SPARQL-Endpoints (Slides)." 15th International World Wide Web Conference (WWW2006), Edinburgh, Scotland, May 2006.

Bizer, C. & Seaborne, A., "D2RQ-Treating Non-RDF Databases as Virtual RDF Graphs", Proceedings of the 3rd International Semantic Web Conference, 2004.

Bloodgood J. M. & Salisbury, M.D., "Understanding the influence of organizational change strategies on information technology and knowledge management strategies." *Decis. Support Syst.* 31, 1 (May 2001), 55-69.

Boffi, P., Bratovich, R., Persia, F., Barberis, A., Martinelli, M., Basso, A., Bucca, G., Nicchio, A. & Bocciolone, M., "Fiber Sensor for Collector Strain Monitoring in the Pantograph-Catenary Interaction," in Optical Fiber Sensors, OSA Technical Digest (CD) (Optical Society of America, 2006), paper TuE45.

Bolloju, N., Khalifa, M. & Turban.E., "Integrating knowledge management into enterprise environments for the next generation decision support." *Decision. Support Systems.* 33, 2 (June 2002), 163-176."

Borgida, A. & Serafini, L., "Distributed Description Logics: Directed Domain Correspondences in Federated Information Sources", in Proc. CoopIS/DOA/ODBASE, 2002, pp.36-53.

Borst, P., Akkermans, H., Top, J., "Engineering ontologies," International Journal of Human-Computer Studies, Volume 46, Issues 2-3, February 1997, Pages 365-406, ISSN 1071-5819, DOI: 10.1006/ijhc.1996.0096.

Brachman, R. J. & Levesque H., J., "Knowledge Representation and Reasoning" *The Morgan Kaufmann Series in Artificial Intelligence*, Morgan Kaufmann, Elsevier, 2004.

Brachman, R. J. & Levesque H., J., "Competence in knowledge representation," In proceedings of the second national conference on artificial intelligence. Pages 189 – 192, AAAI, 1982.

Brachman, R., J., Pigman, V., Hector, G., & Levesque, H. J., “An essential hybrid reasoning system: knowledge and symbol level accounts of KRYPTON,” In Proceedings of the 9th International Joint Conference on Artificial Intelligence, 1985, pages 532—539.

Broekstra, J., Klein, M., Decker, S., Fensel, D., van Harmelen, F. & Horrocks, I., “Enabling knowledge representation on the web by extending RDF schema.” In Proceedings of the tenth World Wide Web conference WWW’10, pages 467–478, May 2001.

C

Campos, J., “Development in the application of ICT in condition monitoring and maintenance,” Computers in Industry, Volume 60, Issue 1, January 2009, Pages 1-20

Cardarella, D.E., "UNIX concepts and capabilities," Electrical Engineering Problems in the Rubber and Plastics Industries, 1990., IEEE Conference Record of 1990 Forty-Second Annual Conference of , vol., no., pp.1-3, 30 Apr-1 May 1990.

Carroll J.J., Dickinson I., Dollin C., Reynolds D., Seaborne A., & Wilkinson K., “Jena: Implementing the semantic web recommendations.” In: Proceedings of the 13th international World Wide Web conference on alternate track papers and posters, 2004, pp. 74–83.

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R., “What Are Ontologies, and Why Do We Need Them?” IEEE Intelligent Systems 14, 1 (Jan. 1999), 20-26.

Chen, H., Perich, F., Finin, T., Joshi, A., “SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications,” First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04), Issue Date: August 2004, pp. 258-267.

Cohn, A.G., Bennett, B., Gooday, J., Gotts, N.M., “Qualitative Spatial Representation and Reasoning with the Region Connection Calculus.” GeoInformatica, 1, 275–316, 1997.

Collins, S. R., Navathe, S., Mark. L., “XML schema mappings for heterogeneous database access.” Information and Software Technology, 2002, vol. 44, no. 4, pp. 251-257(7)

Colomb, R., M., “Ontology and the Semantic Web,” IOS Press, 1 Jan 2007 - Computers - 258 pages.

Clark, J., "Comparison of SGML and XML," World Wide Web Consortium Note 15-December-1997.

Cristani, M., Cuel, R., “A Survey on Ontology Creation Methodologies.” Int. J. Semantic Web Inf. Syst. 1(2): 49-69 (2005).

Cruz, I. F. & Rajendran, A., “Semantic Data Integration in Hierarchical Domains,” *IEEE Intelligent Systems* 18, 2 (Mar. 2003), 66-73.

D

Damiani , E.; David, S.; De Capitani di Vimercati, S.; Fugazza C.; Samarati, P., "Open World Reasoning in Semantics-Aware Access Control: a Preliminary Study," Proceedings of SWAP 2005, the 2nd Italian Semantic Web Workshop, Trento, Italy, December 14-16, 2005.

Davis, E. M., Harding, W. E., Schwartz, R. S., & Corning, J. J., "Solid logic technology: versatile, high-performance microelectronics." *IBM J. Res. Dev.* 8, 2 (Apr. 1964).

Doerr, M., Hunter, J., Lagoze, C. "Towards a core ontology for information integration", Journal of Digital Information, Vol 4, No 1 (2003)

De Giacomo, G. & Lenzerini, M., "TBox and ABox Reasoning in Expressive Description Logics (1996)" in Proc. of KR-96.

Dell'Orco, M., "Intelligent Decision Support Tools For Optimal Planning Of Rail Track Maintenance," 2003.

Dey A,. K., "Understanding and Using Context," *Personal and Ubiquitous Computing*, Volume 5, Issue 1, Pages: 4 - 7, 2001.

Doran, P., "Ontology Reuse via Ontology Modularisation." In Proceedings of KnowledgeWeb PhD Symposium 2006 (KWEPSY2006). 17th June 2006. Budva, Montenegro.

Dzbor, M., Paralic, J., Paralic, M., "Knowledge Management in a Distributed Organisation," in Proceedings of 4 IEEE/IFIP International Conference on Information Technology for Balanced Automation Systems in Productions and Transport, Berlin, Germany, Sept. 2000, pp. 339-348. 115.

Cheryl D., & Severin, G., "Syntactic and Semantic Understanding of Conceptual Data Models" (2001). ICIS 2001 Proceedings. Paper 13.

E

El-Azhary, E. S., Edrees, A., & Rafea, A., Diagnostic expert system using nonmonotonic reasoning. Expert Systems with Applications, 23(2), 137–144, 2002.

Elenius, D., Martin, D., Ford, R., Denker, G., “Reasoning about Resources and Hierarchical Tasks Using OWL and SWRL,” International Semantic Web Conference, 2009.

Elphick, J., “Engineers’ Workbench Closeout Report.” July 2004.

Etherington, D.W., Kraus, S., & Perlis, D., “Nonmonotonicity and the Scope of Reasoning.” Technical report. UMI Order Number: CS-TR-2457, University of Maryland at College Park, 1990.

European Rail Research Advisory Council (ERRAC). “Strategic Rail Research Agenda”, 2002, Published on http://www.errac.org/.

European Commission White Paper – “European transport policy for 2010: time to decide,” 2001.

F

Fallside, D., C. & Walmslet, P., “XML Schema Part 0: Primer Second Edition,” W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-0/.

Feng, J., Smith, J., Wu, Q. et al., “Condition Assessment of Power System Apparatuses Using Ontology Systems.” In: Proc. of Transmission and Distribution Conference, IEEE, Dalin, 2005.

Yu-Hong Feng, Teck-Hou Teng, & Ah-Hwee Tan., “Modelling situation awareness for Context-aware Decision Support.” Expert Syst. Appl. 36, 1 (January 2009), 455-463.

Fuchs, F., Henrici, S., Pirker, M., Berger, M., Langer, G., & Seitz, C., “Towards semantics-based monitoring of large-scale industrial systems.” In: Proceedings of the international conference on computational intelligence for modelling control and automation and international conference on intelligent agents web technologies and international Commerce, CIMCA, IEEE Computer Society, 2006.

Fuchs, F., Hochstatter, I., Krause, M., Berger, M., “A Meta–Model Approach to Context Information,” *2nd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea05, PerCom05)*, Hawaii, USA, March 2005.

G

Gabbay, D., M., Hogger, C.,J., Robinson, J.A., “Handbook of Logic in Artificial Intelligence and Logic Programming: Vols I-V.” Oxford University press, Oxford, 1993-1998.

Gangemi, A., “Ontology Design Patterns for Semantic Web Content,” Proceedings of the Fourth International Semantic Web Conference, Springer, 2005.

Garcia Marquez, F., Lewis, R., Tobias, A., Roberts, C., “Life cycle costs for railway condition monitoring,” accepted for publication in Transportation Research: Part E: Logistics and Transportation Review, 2008, 44(6), 1175-1187.

Gennari, J.H., Musen, M. A., Fergerson, R.W., Grosso, W.E., Crubezy, M., Eriksson, H., Noy, N. F. & Tu. S.W. “The evolution of Protege: an environment for knowledge-based systems development.” *Int. J. Hum.-Comput. Stud.* 58, 1 (January 2003), 89-123.

George, D., "Understanding Structural and Semantic Heterogeneity in the Context of Database Schema Integration", Journal of the Department of Computing, University of Central Lancashire UCLAN, 2005. Preston UK, no. 4.

Gregg, D. G., Goul, M., & Philippakis, A. “Distributing decision support systems on the WWW: the verification of a DSS metadata model.” *Decision Support Systems.* 32, 3 (January 2002), 233-245.

B. N. Grosof, I. Horrocks, R. Volz, & S. Decker., “Description Logic Programs: Combining Logic Programs with Description Logic.” In Proc. of the Twelfth International World Wide Web Conference (WWW 2003), pages 48-57. ACM, 2003.

Gruber, T.R., “Toward principles for the design of ontologies used for knowledge sharing.” Int. J. Hum. Comput. Stud. 43, 5/6 (1995), 907–928.

Guarino, N. & Giaretta, P., “Ontologies and knowledge bases: Towards a terminological clarification.” In Mars, N., editor, Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995, pages 25-32, Amsterdam, NL. IOS Press.

Guarino. N., “Understanding, building and using ontologies.” *International Journal of Human Computer Studies*, 46:293–310, 1997.

H

Hayes, P.J., “The Logic of Frames”, Frame Conceptions and Text Understanding, pp 46-61, Walter de Gruyter and Co. 1979.

Hayes, P.J., “RDF model theory.” W3C Recommendation, 10 February 2004.

Harold, E.,R. & Means, S., “XML in a Nutshell, A Desktop Quick Reference,” January 2001. ISBN 0-596-00292-0.

V. Haarslev & R. Möller. “Racer: An OWL Reasoning Agent for the Semantic Web.” In Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence, Halifax, Canada, October 13, pages 91–95, 2003.

Henning, U., Shingler, R., Umiliacchi, P., Langer, G., “Managing traction data according to a standardised approach: the InteGRail project,” RTS 2007 (The Third International Conference on Railway Traction Systems), Tokyo, Japan.

Holsapple, C. W., Park, S., Stansifer, R. D., & Winston, A.B. “Flexible user interface decision support systems.” In *Proceedings of the Twenty-First Annual Hawaii International Conference on Decision Support and Knowledge Based Systems Track*, Benn R. Konsynski (Ed.). IEEE Computer Society Press, Los Alamitos, CA, USA, 217-222. 1998.

Horrocks, I., “Optimising Tableaux Decision Procedures for Description Logics.” PhD thesis, University of Manchester, 1997.

Horrocks, I., “Ontologies and the semantic web.” Communications of the ACM, 51(12):58-67, December 2008.

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., “SWRL: A Semantic Web Rule Language Combining OWL and RuleML,” W3C Member Submission, 21 May 2004.

Horrocks, I.; Li, L.; Turi, D.; Bechhofer, S., “The Instance Store: DL Reasoning with Large Numbers of Individuals,” *Proceedings of the 2004 International Conference on Description Logics*, Whistler, British Columbia, Canada, June 6-8, 2004.

Horrocks, I.; Patel-Schneider, P., van Harmelen, F., "From SHIQ and RDF to OWL: The making of a Web Ontology Language". Web Semantics: Science, Services and Agents on the World Wide Web 1: 7–26. 2003.

Horrocks, I.; Patel-Schneider, P., McGuinness, D. L., & Welty, C. A., “OWL: a Description Logic Based Ontology Language for the Semantic Web.” In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition), chapter 14. Cambridge University Press, 2007.

Horrocks, I., & Sattler. U., “A Description Logic with Transitive and Inverse Roles and Role Hierarchies.” J. of Logic and Computation, 9(3):385-410, 1999.

Horrocks, I., & Sattler. U., “Ontology Reasoning in the SHOQ(D) Description Logic.” In Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), pages 199-204. Morgan Kaufmann, Los Altos, 2001

Horrocks, I., & Sattler. U., “A Tableau Decision Procedure for SHOIQ. Journal of Automated Reasoning,” 39(3):249-276, 2007.

Horrocks I., Sattler U., and Tobies S., “A Pspace Algorithm for Deciding ALCNIR Satisfiability.” Technical Report. Rheinisci-Westfalische Technische Hochschule.

Horrocks I., Sattler U., & Tobies S., “Reasoning with Individuals for the Description Logic SHIQ.” In David McAllester, editor, Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000), volume 1831 of Lecture Notes in Computer Science, pages 482-496. Springer, 2000.

Hunter, A. & Liu, W**.**, “Measuring the quality of uncertain information using possibilistic logic”,. *Proceedings of the Eighth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU05)*:415-426. LNAI 3571, Springer, 2005.

I

Ide, N., & Pustejovsky, J., “What Does Interoperability Mean, anyway?” Toward an Operational Definition of Interoperability. Proceedings of the Second International Conference on Global Interoperability for Language Resources (ICGL 2010), Hong Kong, China.

ISO 13374-2:2007, “Condition monitoring and diagnostics of machines. Data processing, communication and presentation. Data processing,” Geneva, Switzerland: International Organization for Standardization.

ISO13374 Standard Condition monitoring and diagnostics of machines -- Data processing, communication and presentation -- Part 1: General guidelines http://www.iso.org/

ISO 10303-11:1994, “Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 11: Description Methods: The EXPRESS Language Reference Manual,” Geneva, Switzerland: International Organization for Standardization.

ISO 10303-55:2005, “Industrial automation systems and integration -- Product data representation and exchange,” Part 55: Integrated generic resource: Procedural and hybrid representation, www.iso.org.

ISO15926 Standard - Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 1: Overview and fundamental principles. http://www.iso.org/

ISO15926: Industrial automation systems and integration -- Integration of life-cycle data for process plants including oil and gas production facilities -- Part 2: Data model

ISO10303-21 Industrial automation systems and integration -- Product data representation and exchange -- Part 21: Implementation methods: Clear text encoding of the exchange structure, http://www.iso.org/

Israel, D.J., "The Role of Logic in Knowledge Representation," Computer , vol.16, no.10, pp.37,41, Oct. 1983

J

Joseki. “A SPARQL Server for Jena.” Available from http://www.joseki.org/. 2003

Klinov, P., “Pronto: A non-monotonic probabilistic description logic reasoner.” In: Proceedings of the fifth European semantic web conference, ESWC, 2008.

Meech, A., “Business Rules Using OWL and SWRLTMRF.” e-Book. Advances in Semantic Computing, (Eds. Joshi, Boley & Akerkar), Vol. 2, pp 23 - 31, 2010.

K

Kesich, J. & Golby, A., “MNR Wheel Impact Load Detection – Improved Performance at Reduced Cost,” American Public Transportation Association, Rail Conference 2011, Boston USA.

Klinov, P., “Pronto: A Non-monotonic Probabilistic Description Logic Reasoner.” ESWC 2008: 822-826

Klyne, G., Carroll, J. & McBride, B., “Resource Description Framework (RDF): Concepts and Abstract Syntax,” W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-concepts/.

Kokar, M. M., Matheus, C., J. Baclawski, K., Jerzy A., Letkowski J. A., Hinman, M. & Salerno,J. "Use Cases for Ontologies in Information Fusion," Lecture Notes in Economics and Mathematical Systems Series, 1995, Springer Verlag.

L

Lacy, L., "OWL: Representing Information Using the Web Ontology Language," Trafford, 2005.

Lammari, N., Matais E., "Building and maintaining ontologies: a set of algorithms," *Data and Knowledge Engineering*, Volume 48, Issue 2, pp155-176, 2004.

Lakshmanan, L.V.S., Sadri, F., & Subramanian, I. N., "Logic and algebraic languages for interoperability in multidatabase systems." Technical report, Concordia University, Montreal, Feb 1996. Accepted to the Journal of Logic Programming.

Lee, J. & Goodwin, R., "Ontology management for large-scale enterprise systems," Electronic Commerce Research and Applications 5, (2006) 2–15.

Lehrasab, N. Dassanayake, H. P. B., Roberts, C., Fararooy, S., & Goodman, C. J. "Industrial fault diagnosis: pneumatic train door case study," *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*. Volume 216, Number 3 / 2002, pp175-183.

Lewis, J. W., "Medical information systems integration." (Panel Discussion),
ACM 81: Proceedings of the ACM '81 conference, 1981.

Lewis, R., & Roberts, C., “Using non-monotonic reasoning to manage uncertainty in railway asset diagnostics,” Expert Systems with Applications 01/2010.

Liebowitz, J., “Expert System: A Short Introduction”, Engineering Fracture Mechanics, 50(5/6), 601-607,1995.

Liebowitz, J., “Knowledge management and its link to artificial intelligence,” Expert Systems with Applications, Volume 20, Issue 1, January 2001, Pages 1-6, ISSN 095, DOI: 10.1016/S0957-4174(00)00044-0.

Lukasiewicz, T. & Straccia, U., “Managing Uncertainty and Vagueness in Description Logics for the Semantic Web.” In Journal of Web Semantics, 2008.

Lutz, C., Wolter, F., & Zakharyaschev, M., “Temporal Description Logics: A Survey.” In Proceedings of the 2008 15th International Symposium on Temporal Representation and Reasoning (TIME '08). IEEE Computer Society, Washington, DC, USA, 3-14.

M

Maly, T. & Scheinzer, “Measurement Data Treatment in Multi-Sensor Applications for Railway Vehicle Inspection.” Journal of Physics: Conference Series 13, 2005.

Manola, F., Miller, E., & McBride, B., “RDF primer.” W3C Recommendation, 2004.

McArthur, S.D.J., Davidson, E.M., &Catterson, V.M., "Building multi-agent systems for power engineering applications," Power Engineering Society General Meeting, 2006. IEEE , vol., no., pp.7.

McIlraith, S., Son, T. C., & Honglei, Z., "Semantic Web Services," *Intelligent Systems* (IEEE) **16** (2): 46–53, March 2001.

Minsky, M. "A framework for representing knowledge." In Mind Design (editor Haugeland, J), pages 95 – 128, MIT press Cambridge, MA. 1982.

Miguelanez, E., Brown, K., Lane, D., Lewis, R. & Roberts, C. "Fault diagnosis of a train door system based on the semantic knowledge representation", In Proceedings of the 4th IET International Conference on Railway Condition Monitoring, Derby, UK.

Motik, B., Cuenca Grau, B., Horrocks, I., & Sattler, U., "Representing ontologies using description logics, description graphs, and rules." Artificial Intelligence, Volume 173, Issue 14, September 2009, Pages 1275-1309, ISSN 0004-3702.

Motik, B., Sattler, U., & Studer, R., "Query Answering for OWL-DL with Rules." The Semantic Web – ISWC 2004, Lecture Notes in Computer Science Volume 3298, 2004, pp 549-563.

Motik, B., Horrocks, I., Rosati, R., & Sattler, U., "Can OWL and Logic Programming Live Together Happily Ever After?" The Semantic Web - ISWC 2006
Lecture Notes in Computer Science Volume 4273, 2006, pp 501-514.

N

Nash, A., Huerlimann, D., Schuette, J. & Krauss, V.P. “RailML - a standard data interface for railroad applications,” Computers in Railways IX, WIT Press, Ashurst Lodge, Ashurst, Southampton, 2004.

Noy, N. F. & McGuinness, D. L., "Ontology Development 101: A Guide to Creating Your First Ontology," *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, March 2001.

Network Rail, Business Process Document, Control of Wheel Impact Forces, NR/SP/TRK/0133 – June 2006.

O

Ollier, B,. “Intelligent Infrastructure the Business Challenge”, Railway Condition Monitoring, 2006. The Institution of Engineering and Technology International Conference on.

Obitko M., & Smid, J., “Ontology Design with Formal Concept Analysis,” Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications, Ostrava, Czech Republic, September 23-24, 2004.

P

L. Padgham & P. Lambrix. “A framework for part–of hierarchies in terminological logics.” In J. Doyle, E. Sandewall, and P. Torasso, editors, Principals of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR’94), pages 485–496.Morgan-Kaufmann, 1994.

Paschke, A., & Boley, H., “Rule Markup Languages and Semantic Web Rule Languages.” In A. Giurca, D. Gasevic, & K. Taveter (Eds.) Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches (pp. 1-24). Hershey, PA: Information Science Reference, 2009.

Patel-Schneider, P. F., Brachman, R. J. & Levesque H. J. “ ARGON: Knowledge representation meets information retrieval,” In Proc. of the 1st Conf. on Artificial Inteligence Applications (1984).

Patel-Schneider, P. F., & Horrocks I., “OWL Web Ontology Language Semantics and Abstract Syntax Section 3.” Direct Model-Theoretic Semantics, 2004. http://www.w3.org/TR/2004/REC-owl-semantics-20040210/direct.html#3.4.

Pedersen, M. & Larsen, M. H., “Distributed knowledge management based on product state models - the case of decision support in health care administration.” *Decis. Support Syst.* 31, 1 (May 2001), 139-158.

Pinto, H., S. “Ontologies: How can they be built?” Knowledge and Information Systems, Springer, July 1, 2004.

Prendergast, K., "Condition monitoring on the Class 390 Pendolino," Railway Condition Monitoring, 2008 4th IET International Conference on , vol., no., pp.1-6, 18-20 June 2008.

Prud'hommeaux, E., & Seaborne, A., "SPARQL query language for RDF. W3C Candidate Recommendation," 2007, Available from http://www.w3.org/TR/rdf-sparqlquery/.

Q

Quillian. M., R., "Semantic Memory," Semantic Information Processing, pages 216 – 270, MIT Press Cambridge, MA, 1968.

R

"RDF Vocabulary Description Language 1.0: RDF Schema," 10 February 2004. http://www.w3.org/TR/rdf-schema/.

Rector. A., "Modularisation of Domain Ontologies Implemented in Description Logics and related formalisms including OWL." in Knowledge Capture 2003, (Sanibel Island, FL, 2003), ACM, 121-128.

Rector, A., Horridge, M., Iannone L., & Drummond, N., "Use Cases for Building OWL Ontologies as Modules: Localizing, Ontology and Programming Interfaces & Extensions". 4th International Workshop on Semantic Web Enabled Software Engineering (SWESE 2008), Karlsruhe, Germany.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H & Wroe, C., “OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors \& Common Patterns,” 14th International Conference on Knowledge Engineering and Knowledge Management EKAW 2004, 5-8th October 2004 - Whittlebury Hall, Northamptonshire, UK

Roberts C., Dassanayake H.P.B., Lehrasab N., & Goodman C.J., “Distributed quantitative and qualitative fault diagnosis: railway junction case study.” Control Engineering Practice, Volume 10, Number 4, April 2002 , pp. 419-429(11).

Roberts, C., Lewis, R. & Amoore, J. “Making the case for railway condition monitoring” (In Proceedings of the 7th World Congress of Railway Research, Montreal, Canada, 2006).

Routeledge, R., Bird, L., Goodchild, A., “UML and XML schema,” Australian Computer Science Communications archive, Volume 24 Issue 2, January-February 2002, Pages 157 - 166

Rubenstein-Montano, B., Liebowitz, J., Buchwalter, J., McCaw, D., Newman, B., & Rebeck, K, “A systems thinking framework for knowledge management.” *Decis. Support Syst.* 31, 1 (May 2001), 5-16.

S

Sandnes, F. E., Zhang, Y., Rong, C., & Tianruo Yang L., “Ubiquitous Intelligence and Computing.” 5th International Conference, UIC 2008, Oslo, Norway, June 23-25, 2008 Proceedings, Springer, 16 Jun 2008.

Samad, S., McLaughlin, P., & Lu, J., “System architecture for process automation: Review and trends, Journal of Process Control,” Volume 17, Issue 3, Special Issue ADCHEM 2006 Symposium, March 2007, Pages 191-201, ISSN 095..., DOI: 10.1016/j.jprocont.2006.10.010.

Saha, D. Mukherjee, A. & Bandopadhyay, S., “Networking Infrastructure for Pervasive Computing:Enabling Technologies and Systems,” Kluwer Academic, 2002.

Schild, K., “A Correspondence Theory for Terminological Logics: Preliminary Report,” In Proc. of IJCAI-91, 1991, pages 466-471.

Seidenberg, J. & Rector, A., “Representing Transitive Propagation in OWL”,
Proc. of the 25th Int. Conf. on Conceptual Modeling (ER 2006), volume 4215 of LNCS.

Shadbolt, N., Hall, W., & Berners-Lee, T., "The semantic Web revisited," *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol.21, no.3pp. 96- 101, Jan.-Feb. 2006.

Shingler, R., & Umiliacchi, P., “Advances in railways maintenance: the EuRoMain project”, WCRR 2003 - World Conference on Railway Research, Edinburgh, 2003.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y., “Pellet: A practical OWL-DL reasoner. Web Semantics.” 5, 2 (Jun. 2007), 51-53.

Smith M. K., Welty, C., & McGuinness, D. L., “OWL Web Ontology Language Guide,” W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-guide.

Staab, S., Erdmann, M., & Maedche, A., “Engineering Ontologies Using Semantic Patterns.” In Procs. IJCAI-01 Workshop on E-Business & the Intelligent Web, 2001.

Staab, S. & Studer, R., “Handbook on Ontologies,” International Handbooks on Information Systems, 2010, Springer.

Stroud, G. & Elphick, J. “Remote condition monitoring - an asset management viewpoint,” Railway Condition Monitoring: Why? What? How? 2005 IEE Seminar on (Ref. No. 2005/10813) Volume, Issue , 23-23 Feb. 2005 Page(s):0_12 - 3/8

Sequada, J., “Introduction to: Open World Assumption vs Closed World Assumption,” http://semanticweb.com/ Articles, Insight, Introduction to, Learning, Standards, 2012.

Shim, J. P., Warkentin, M., Courtney, J.F., Power, D.J., Sharda, R. & Carlsson, C., “Past, present, and future of decision support technology.” *Decis. Support Syst.* 33, 2 (June 2002), 111-126.

Shingler, R. & Umiliacchi, P., “Advances in railways maintenance: the EuRoMain project,” WCRR '03–World Conference on Railway Research, 2003.

Siever, E., Figgins, S. & Weber, A., “Linux in a Nutshell, A desktop Quick Reference,” O’Reilly 2003, ISBN 0-596-00482-6.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y., “Pellet: A practical OWL-DL reasoner.” Web Semant, 5, 51–53. 2007.

SPARQL “Sparql: Query Language for RDF,” W3C Candidate Recommendation 2007, http://www.w3.org/TR/rdf-sparql-query/.

Spyns, P., Meersman, R., & Jarrar, M. 2002., “Data modelling versus ontology engineering.” *SIGMOD Rec.* 31, 4 (Dec. 2002), 12-17.

Strong, D.M., Lee, Y.W. & Wang. R.Y., “Data quality in context.” *Commun. ACM* 40, 5 (May 1997), 103-110.

Ba, S., Lang, K., R., & Whinston, A., B., “Enterprise decision support using Intranet technology,” Decision Support Systems, Volume 20, Issue 2, June 1997, Pages 99-134, ISSN 016, DOI: 10.1016/S0167-9236(96)00068-1.

T

Thurston, M.G., “An open standard for Web-based condition-based maintenance

systems," in: 2001 IEEE Autotestcon Proceedings. IEEE Systems Readiness Technology Conference, 2001, pp. 401–415.

Thurston, M., & Lebold, M., "Open Standards for Condition-Based Maintenance and Prognostic Systems," MARCON'01, Gatlinburg, USA, May 6-9, 2001.

Trastour, D., Preist, C., & Coleman, D., "Using Semantic Web Technology to Enhance Current Business-to-Business Integration Approaches," In *Proceedings of the 7th international Conference on Enterprise Distributed Object Computing* (September 16 - 19, 2003). EDOC. IEEE Computer Society, Washington, DC, 222.

Tsarkov D., & Horrocks, I., "DL Reasoner vs. First-Order Prover," In Proc. of the 2003 Description Logic Workshop (DL 2003), volume 81 of CEUR 2003, pages 152-159.

Tudorache, T., Noy, N., F., Tu, S., & Musen, M. A., "Supporting Collaborative Ontology Development in Protégé," The Semantic Web - ISWC 2008
Lecture Notes in Computer Science Volume 5318, 2008, pp 17-32.

U

Umiliacchi, P., Shingler, R., Langer, G., & Henning, U. "A new approach to optimisation through intelligent integration of railway systems: the InteGRail project." In: SeventhWorld Congress on Railway Research (WCRR), Montreal, Canada, 2007.

Umiliacchi, P., Lane, D., & Romano, F., “Predictive maintenance of railway subsystems using an Ontology based modelling approach,” In: Ninth World Congress on Railway Research (WCRR), Lille France, 2011.

Uschold, M., & Jasper A., “Framework for Understanding and Classifying Ontology Applications,” Computer and Information Science, Methods, Volume 18, pp1-12, 1999.

V

Vardi, M. Y., “Why Is Modal Logic So Robustly Decidable?” In N. Immerman and P. Kolaitis, editors, Proc. of a DIMACS Workshop on Descriptive Complexity and Finite Models, volume 31 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 149–184, Princeton University, USA, January 14–17 1996. American Mathematical Society.

van Harmelen, F., Lifschitz V., & Porter B., (Editors) “Handbook of Knowledge Representation,” © 2008 Elsevier B.V. All rights reserved DOI: 10.1016/S1574-6526(07)03003-9

Valente, A.; Holmes, D.; & Alvidrez, F.C., “Using a Military Information Ontology to Build Semantic Architecture Models for Airspace Systems,” Aerospace Conference, 2005 IEEE , vol., no., pp.1-7, 5-12 March 2005.

Verstichel, S., Strobbe, M., Simoens, P., De Turck, F., Dhoedt, B., & Demeester, P. (2008). “Distributed reasoning for context-aware services through design of an

OWL meta-model.” In: Fourth international conference on autonomic and autonomous systems, ICAS (pp. 70–75).

W

Wallace, D., “How to put SCADA on the Internet,” Control Engineering, Sept 2003.

West, M., Developing High Quality Data Models, Morgan Kauffman, 2011. Elsevier Inc. ISBN 978-0-12-375106-5.

Wolstencroft, K.,Brass, A., Horrocks, I., Lord, P., Sattler, U.; Turi, D., & Stevens, R. 2005. “A Little Semantic Web Goes a Long Way in Biology.” In Proc. of ISWC-4, number 3729 in LNCS, 786–800. Springer.

X

Y

Yen, D. C., Huang, S., & Ku, C., “The impact and implementation of XML on business-to-business commerce,” *Comput. Stand. Interfaces* 24, 4 (Sep. 2002), 347-362.

Z

Zemánek, J. & Schenk, S., “Optimizing SPARQL Queries over Disparate RDF Data Sources through Distributed Semi-Joins.” In Proceedings of International Semantic Web Conference (Posters & Demos), 2008.

Zhou, F.N., Archer, N J., Bowles, J M., Duta, M., Henry, M. P., Tombs, M. S., Zamora, M.E., Baker, S., & Burton, C., “Remote condition monitoring and validation of railway points," IEE Computing & Control Engineering Journal, pp. 221-230, October 2002.