Transportation Research Part C 19 (2011) 617–643

Contents lists available at ScienceDirect

Transportation Research Part C

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

Efficient data integration in the railway domain through an ontology-based methodology

Stijn Verstichel [a,*], Femke Ongenae [a], Leanneke Loeve [b,2], Frederik Vermeulen [c,1,3], Pieter Dings [b,2], Bart Dhoedt [a], Tom Dhaene [a], Filip De Turck [a]

[a] Ghent University – IBBT, Department of Information Technology, Broadband Communication Networks, G. Crommenlaan 8/201, 9050 Gent, Belgium
[b] DeltaRail, PB 8125, 3503 RC Utrecht, The Netherlands
[c] Televic NV, Leo Bekaertlaan 1, 8870 Izegem, Belgium

ARTICLE INFO

Article history:
Received 29 June 2009
Received in revised form 18 October 2010
Accepted 19 October 2010

Keywords:
Ontology
Information integration
Railway
Network Statement Checker
InteGRail

ABSTRACT

The fragmented and ever more specialized nature of today's railway systems makes it more and more complex to operate. An increasing number of actors are involved in the operation of a railway service. Infrastructure management is being separated from the operational aspect. Apart from the traditional state-owned train operators, open access and private operators start using the same infrastructure as well. Additionally, an increasing number of information systems, such as for real-time passenger information and entertainment need to exchange information. Therefore, Information & Communication Technologies have an increasingly vital role to play in the operation of the railways. However, as the use of stand-alone information systems improves the efficient operation of a single railway stakeholder, due to the complex fragmented nature, there is a clear need to integrate and correlate the available information. A high level of structured interoperability between information systems is required to correctly combine and manage this complex information. Several mechanisms exist to integrate information systems. The approach presented in this paper discusses the integration on data level. The main benefit of this approach is that it supports independent application development. It is after all undesirable and nearly impossible to centralise application development in world-wide fragmented and large systems, such as the railways. We will discuss a number of approaches towards data integration. Two main technologies are considered, namely Unified Modelling Language (UML) and ontologies. An ontology-based solution is compared with an UML-based approach. The advantages and disadvantages of both UML and ontology-based approaches are presented. The results are evaluated by means of a demonstrator developed as part of the InteGRail project (Intelligent Integration of Railway Systems), an FP6 EU research project. We believe that this demonstrator, the Network Statement Checker, is an ideal candidate to demonstrate the advantages of an ontology-based integrated information system. This tool allows the infrastructure operators to combine the network statements of different countries in different formats and to analyse them in a transparent way. The ontology-based approach shows clear advantages compared to the UML approach, by means of the formally defined model, but on the other hand the performance of the currently available tools is still to be improved. However, we believe that the augmented value of an

* Corresponding author. Tel.: +32 9 33 14 900; fax: +32 9 33 14 899.
E-mail address: Stijn.Verstichel@intec.ugent.be (S. Verstichel).
[1] Present address: APT – Track Products and Measurement Devices, Mechelsevest 18/0301, 3000 Leuven, Belgium. Tel.: +32 16 23 20 40; fax: +32 16 23 89 10.
[2] Tel.: +31 30 3005 100; fax: +31 30 3005 150.
[3] Tel.: +32 51 30 30 45; fax: +32 51 31 06 70.

0968-090X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trc.2010.10.003

ontology-based approach is also to be found in lower development costs because of its potential reuse in multiple applications, since their philosophy is to serve as a domain model instead of as a data model for a specific application.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

From an engineering point of view, the railway domain is a large-scale integration challenge. Historically, this has been the case in the mechanical field, for instance wheel and track dimensions. This is witnessed by numerous standards. Unfortunately in Europe with its history of national railway companies, standards are mostly nation-specific. In the field of electrical integration a large body of knowledge has been built up as well. For example electromagnetic compatibility norms and a standardized validation procedure make for a smooth electrical integration.

Industry-wide integration in the information domain is only in its infancy. From an efficiency point of view, this field leaves much room for improvement (as did the integration in the mechanical and electrical domain). A number of potential integration mechanisms exist. Firstly, one could integrate individual applications by means of re-implementing them in one domain-wide application. Secondly, Application Programming Interfaces (API) could be exposed as well offering application developers the possibility to use one another's applications and business information. Lastly, the integration could also be done at data level. This means that the individual applications continue to be developed independently, but a commonly agreed domain model is established to exchange information between the collaborating stakeholders. This last mechanism is elaborated in this paper and more precisely, an ontology-based approach is compared with a UML-based one. In a large world-wide and fragmented domain, such as the railways, we believe the first approach is unfeasible. After all, centralising application development in such an environment would be problematic. The second approach is also not ideal, because of the dependency on the exposed APIs. Even the slightest change in this interface results in a necessary adaptation of numerous other applications, since many stakeholders might be using this particular API. The integration by means of commonly agreed domain models is therefore a better suited approach. Additionally, by using ontologies as a means of constructing these models, the additional benefit of formally defined and description logics supported models is exploited. It does not only define the syntax of the information exchanged, but also the semantics of that information, by using a shared and extendable domain-wide model.

Information integration is not exclusively a challenge for the railway domain. The TeleManagement Forum (TM Forum), for example, is a consortium in the telecom industry. It consists of over 800 partners (Network Operators, Service Providers, Equipment Suppliers, Software Suppliers, and System Integrators) and provides a set of recommendations and guidelines for building efficient network and service management applications in telecom oriented companies. Their focus is on telecom business process modelling and models for data exchange between different IT systems in telecommunication. The main goal is the standardized interaction between different telecom stakeholders and optimized internal organization in telecom companies. Important examples of telecom business processes are: *DSL fulfilment, Repair Management and Bill Invoice, Payments and Inquiry Management*.

An important outcome of TM Forum is Shared Information Data (SID), which is an information and data model. It describes how to model telecom business process information and data and is a common vocabulary that creates a bridge between the telecom business and IT, which then simplifies the integration of IT systems. SID also describes relationships (associations) between entities. The basic building blocks are Aggregate Business Entities (ABEs). Example ABEs in the Resource category are: Resource Specification, Resource Topology, Resource Performance, Resource Usage, *etc*. Example ABEs in the Customer category: Customer Order, Customer Problem, Customer Bill, Customer Bill Inquiry, *etc*. Unified Modelling Language (UML) is used to describe the different ABEs and the relations (Reilly and Wilmes, 2008). A number of integration tools exist which allow semantic data integration through visual programming.

For wider specifications of system integration, as is the case in the railway domain, the preferred solution for the integration of data is one that avoids any major alteration to existing system design. Further to this, there is a requirement to enable systems to provide data to other stakeholders that can be extended at any time without major redesign. This interaction can be achieved through the implementation of a common vocabulary that forms the foundation for communication between applications. In this paper, we propose an underlying technical framework and methodology to realize the data integration in the railway domain. Section 2 introduces the InteGRail approach towards information integration. InteGRail was an FP6 European Rail Research project. Section 3 describes the advantages and disadvantages of both a UML – as well as an ontology-based integration approach, while Section 4 presents a number of methodologies to construct information integration ontology models. By means of a demonstrator, we evaluate in Section 5 the performance of the ontology-based methodology for an important railway use case. In Section 6, we present our views on how to bring the idea of an ontology-based information platform forward towards an industrial deployment. Section 7 concludes this paper.

2. Data integration to improve business opportunities and efficiency

Data and information integration in the field of the railway (and also the intermodal) transportation is the subject of several ongoing and finished European Research projects. Their goal is to define information standards to improve the efficiency

of the industry. At the communication protocol level, the Internet Protocol (IP) has become the de facto standard. At the syntactical level, the Euromain project (EuRoMain Consortium, 2003) has standardized towards XML and has proposed schemas for a limited number of cases of data integration. While this approach reaches the goal of syntactical integration, the need exists to have on top of that a semantic integration. The syntactical integration has no answer for the integration of existing datasets in the same information domain that happen to have a semantic mismatch, for example the synonym problem as defined further in this paper. A transformation step is needed in this case. For the integration of datasets extending the information set currently defined in schemas, standardization effort is needed before this data can be used. These examples show that in the current railway state of the art a new interface definition is needed for each new usage of the information. It is the semantic mismatch – or undefinedness – that obstructs the single specification of information.

Many railway undertakings nowadays encounter the boundaries of their proprietary isolated information systems. There is a growing need to integrate these systems in order to correlate the information. Firstly, the integration of this data augments its value drastically, for example when sensor measurement data is augmented with its exact time, location or measurement accuracy. Secondly, the correlation aspect improves the handling of the enormous amounts of data readily available today. Moreover, due to the amounts of data available, there is a need to adopt a well-structured approach. At the same time, the reduction of the costs to maintain the systems and assets is of great importance in the current environment of increasing competition. Further cost reduction is limited because of the lack of knowledge of the world outside these systems. These islands of systems often contain duplicated data, counterproductive for the optimization of business processes.

To augment the value of the available data, semantic context information should be attached to it. In the example given in Fig. 1a, this annotation is the addition of the time and place to the raw measurement. This is the second step in the process to create intelligence out of the raw data, as generally accepted in the Semantic Web world; the first step being the exposure of raw data without any annotation at all. A possible third step is to create knowledge out of the information available. This means that the systems using the semantically annotated information should be thought how the augmented information should be used. The ultimate goal is to create a self-organizing system, which does not only understand the data, but also has the intelligence to understand when to use the knowledge. Fig. 1b shows the process with a concrete example illustrating the differences in every step in the chain. The raw data is just a measurement, while the information adds the place and time to it as context information. The knowledge in the third step represents the domain rules that need to be followed in case of such a measurement, while the last step indicates the situation when further processing of this information is not necessary anymore.

The InteGRail approach (InteGRail Consortium, 2008) currently developed has to be positioned on the border between step 2 and step 3, with the emphasis on augmenting the data with context so that a structured and meaningful exchange of information between railway stakeholders is supported. As will be indicated in the following paragraphs, example implementations of knowledge injection in the system have been demonstrated in the use cases developed by the consortium.

Not only example domain modelling has been performed within the project, but InteGRail has also specified a modular and generic platform for such systems to cooperate and exchange information. At the end of the project, a Proof-of-Concept platform implementation has been demonstrated. The continued implementation and deployment of a well-developed and moderated InteGRail Service Platform is therefore still necessary to exploit the InteGRail results in a commercial and industrial environment. Such a platform should be a robust, secure and scalable service platform for semantic data exchange and integration.

3. Evaluation of ontology-based data integration

3.1. The data integration problem

Data integration is concerned with unifying data with some common semantics but originating from a set of heterogeneous, distributed and autonomous (unrelated) sources (Buccella et al., 2003). The integration process provides the users

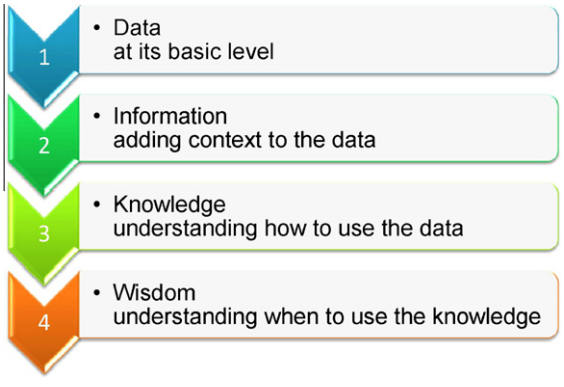

Fig. 1a. How to get from raw data to intelligent wisdom.

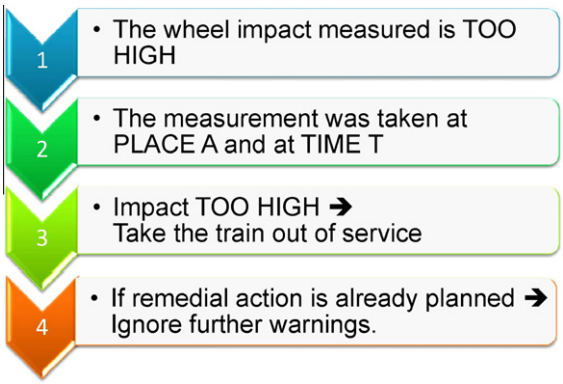

Fig. 1b. Example illustrating the knowledge engineering process.

with a unified view of this data. This data can be distributed on different hosts that are connected through a network. The data sources are autonomous, which means that users and applications can access them through a local or a federated system. On top of that, the data can also exhibit four types of heterogeneity (Cui and O'Brien, 2000):

- **Structural heterogeneity**: The data sources have different data models.
- **Syntax heterogeneity**: The data sources have different languages and data representations.
- **Implementation heterogeneity**: The different data sources run on different hardware and operating systems.
- **Semantic heterogeneity**: The conceptualization of the different data sources is influenced by the designers' view of the concepts and the context to be modelled. The different data sources use different contexts to give meaning to the data.

Semantic heterogeneity can give rise to semantic conflicts between the different data sources. Three types of conflicts can be identified (Goh, 1997). Confounding conflicts occur when data items seem to have the same meaning, but differ in reality for example owing to different temporal contexts. The usage of different reference systems to measure a value causes scaling conflicts, for example gallons versus litres. Naming conflicts take place when the naming schemes of the different information sources differ significantly. Three common problems are:

- **Synonyms**: The concepts are semantically equivalent. Models can use different terms to refer to the same concept or the same properties are modelled differently by different systems.
- **Homonyms**: The concepts are semantically unrelated. The same terms are used for different concepts.
- **Classification**: The concepts are semantically related. For example, a concept is a generalization or a specialization of another concept.

All these characteristics make it difficult to integrate the data. Heterogeneity is the most challenging problem. In order to achieve semantic interoperability in a heterogeneous information system, the meaning of the information that is interchanged has to be understood across the systems. Thus, a common semantic model of all the data in the different sources is required.

3.2. Introduction to the OWL technology

A short, but comprehensive definition of an ontology, based on the definition by Gruber (1993), is: "An ontology is a formal specification of an agreed conceptualization of a domain in the context of knowledge description". An ontology thus describes in a formal commonly agreed manner the concepts in a certain domain, their attributes and their relationships.

Confusion often rises about the difference between ontologies and conceptual schemas. The processes of constructing a conceptual schema and a domain ontology are similar. However, the objective (scope) of these models is very different (Fonseca and Martin, 2007). Conceptual schemas are built with a specific information system in mind. They have the practical purpose of defining, constraining and limiting what knowledge and data is going to be used in a certain information system. Ontologies have as purpose to make all the knowledge about a certain domain explicit. Although ontologies can be used by an information system or to support its development, they are not constructed with this information system in mind. Ontologies can therefore be easily re-used by various information systems or applications.

The most used and well-known language to describe ontologies is Ontology Web Language (OWL) (Horrocks et al., 2003; Bechhofer et al., 2004; McGuinness and van Harmelen, 2004). This technology allows for a common, formally defined and description logics supported data-format to be specified. Effectively, it models the world around the systems in a graph model. This graph contains the concepts present in the modelled domain, the relations between those concepts and potentially

classification axioms which specify generic knowledge about the domain and can be exploited by the business logic of an application at a generic level of abstraction, by means of generic reasoning mechanisms. This common, agreed data-format can then be used to exchange the information and its attached domain model beyond the undertakings' system boundaries, thus facilitating an integrated view of the asset's conditions. Moreover, the ontology technology not only allows to model the data in a formalized manner, but also to reflect the semantics of that data, thus the information and the knowledge of those systems.

Due to the foundation of ontologies in description logics (Nardi et al., 2003), the models and description of the data in these models can be formally proofed. It can also be used to detect inconsistencies in the model as well as infer new information out of the correlation of this data. An example of such inference is root-cause analysis. However, for the latter to be possible, it is paramount that extra effort and care has to be taken to model the system in such a way that this inference is made possible. This is perhaps the most difficult part of the entire task to integrate a system in such a semantic information exchange platform. This proofing and classification process is referred to as reasoning. Due to the supporting paradigm of the ontology constructs being founded in description logics, these reasoners are implemented as generic software modules, independent of the domain-specific problem. The reasoning process can be adopted in a number of ways. On the one hand, it can be used to check the model offline, before deployment in the service platform. On the other hand it can also be used at runtime to classify and realize the data, according to their relationships and the values of these relationships.

An example of such domain knowledge modelled in the ontology instead of in the application is graphically illustrated in Fig. 2. The example shows how the transfer of domain-specific rules from the application to the model can facilitate a reusable and generic end-user application. As the domain logic is incorporated inside the model, the application does not need to be changed when the characteristics of the that logic change. The first approach, illustrated at the top of the diagram, has the domain rules, *i.e.* that a fault is a measurement with a value of 10 in the first system, and a measurement with a value of 20 in the second measurement system, included in the application. Thus, with every other deployment with yet another type of measurement systems, the end-user application would have to be altered as well. In the second approach, illustrated at the bottom of the diagram, the logic is included in the model. Together with a reasoner, the end-user application would not have to be changed for new deployments with other types of measurement systems. This work is shifted towards the model engineering process, a task of domain experts. As a result, the application should only ask for *Faults*. The reasoner can then automatically infer for every domain, using the ontology model, which measurements are actually faults for that specific domain.

OWL has different levels of expressive power. This was motivated by the principle of minimality, not including as many modelling features as possible, but constriction of expressivity to make inference feasible. It consists of three sublanguages, each of them varying in their trade-off between expressiveness and inferential complexity. They are, in order of increasing expressiveness:

- **OWL Lite**: supports classification hierarchies and simple constraint features.
- **OWL DL**: OWL Description Logics, a subset providing great expressiveness without losing computational completeness and decidability.
- **OWL Full**: supports maximum expressiveness and syntactic freedom however without computational guarantees.

Using one of the three sublanguage-flavours of OWL, one can easily adapt to the required expressiveness. Arguably the most interesting sublanguage for many application domains is OWL DL, balancing great expressiveness with inferential efficiency. Due to its foundation in Description Logics, OWL DL is also very flexible and computationally complete. This means that all conclusions are guaranteed to be computable. The decidability of OWL DL, being that all conclusions will be reached in finite time, is an imported aspect as well.

OWL is also compatible with the web architecture. It has, amongst others, an XML-based (Bray et al., 2006a,b) encoding and it is backward compatible with RDF Schema (Guha et al., 2004), which can be seen in Fig. 3. As ontologies are also tai-

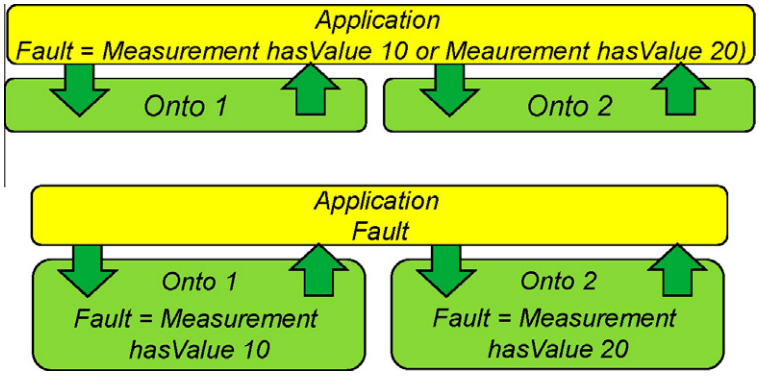

Fig. 2. Example illustrating the use of descriptions and restrictions in an ontology.

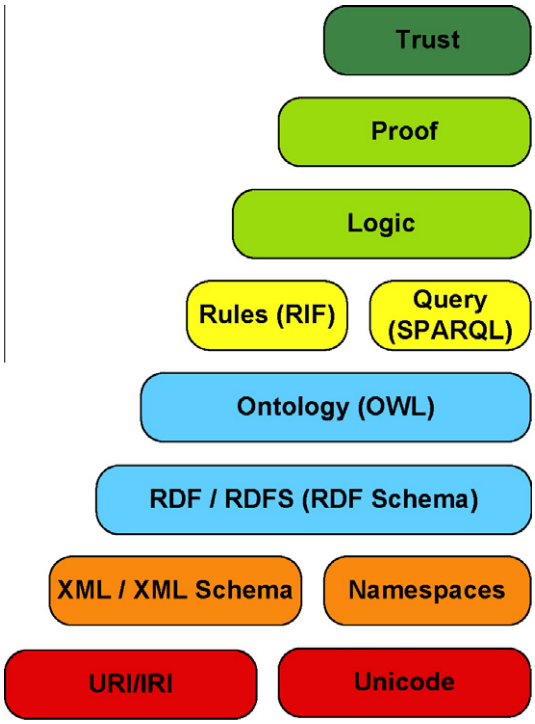

Fig. 3. Layered view on Semantic Web enabling technologies.

lored towards the distributed nature of the Web, OWL additionally provides constructs for (de-)composition, extension, adaptation, sharing and reuse.

3.3. Comparison of OWL and UML

In order to achieve semantic interoperability in a heterogeneous information system, the meaning of the information that is interchanged has to be understood across the systems. Thus, a common semantic model of all the data in the different sources is required. Two languages have been commonly used to express this common semantic model, namely UML (OMG, 2009; OMG, 2007a,b) and OWL.

UML and OWL originated from very different ideas (Hart et al., 2004). UML was designed to integrate competing proposals for modelling languages in the area of software engineering and to promote object-oriented design. It is meant to be used by humans to document and communicate about their software designs. It is a standard of the Object Modelling Group (OMG). OWL, on the other hand, was designed as standard language for the representation of ontologies on the World Wide Web. It was thus meant to be used by systems rather than humans. OWL is a standard recognized by the World Wide Web Consortium (W3C) since 10 February 2004.

These languages also have different notational foundations and views with respect to expressivity. The main notation of UML is in terms of a graphical model rather than a formal language. UML is a combination of different model types each covering a specific aspect of the overall software system, for example class diagrams or sequence diagrams. This decision was driven by allowing as much expressive power as possible. OWL is based on a formal language (Description Logics) and thus has well-founded semantics.

OWL and UML also differ in their ability to express semantics. Some formal constraints of UML can be captured by using Object Constraint Language (OCL) (OMG, 2006). OCL attempts to define formal semantics of parts of the language, for example for class diagrams. Validation and transformation methods exist for these partials semantics, such as RationalRose (Quatrani, 2000), but they are not popular or often used. As there is no reasoning, the user has to define the whole classification tree himself. OCL is also able to express some rules, but OCL has no formal semantics. Additionally, no dedicated query language exists for UML. Because OWL has formal semantics, automated reasoning is possible. The consistency of the model can be checked, classification can be done by the system and new knowledge can be derived by logical inference. This means that the user doesn't have to define the whole classification tree himself. OWL can be easily integrated with different rule platforms. RDF (Beckett and McBride, 2004), of which OWL is an extension, also has a dedicated query language namely SPARQL (Prud'hommeaux and Seaborne, 2008).

As can be seen, the two languages complement each other. UML is designed for model building by human experts, while OWL is designed to be used at runtime to provide guidance for intelligent processing methods. A lot of information is already specified in UML class diagrams, but this format does not allow reasoning and is not compatible with the World Wide Web. Approaches were developed for transforming UML class diagrams into OWL to overcome these problems (Baclawski et al., 2001; Falkovych et al., 2003). UML has also been used as modelling syntax for knowledge representation languages because there was a lack of tools available to visualize and edit ontologies in the past.

3.4. Why use ontologies for data integration

To address the problem of semantic heterogeneity, a common semantic model of all the data in the different sources is required. This makes hidden and implicit knowledge explicit. It includes two necessary steps, namely building the local vocabularies and defining the mappings. Sometimes a third step is included in which a shared vocabulary is built. By using a shared vocabulary, new information sources can be added without need of modification to the other sources or their vocabularies. Only new terms and relations need to be added to the shared vocabulary. This shared vocabulary is not a necessity in case mappings between the different local vocabularies can easily be established in a direct way. Defining direct mappings is however often a difficult task. This is further explained in Section 4.1.

Both UML and ontologies (OWL) could be used to construct the vocabularies. However, as discussed in the previous section, these languages differ a lot from each other. Below it is explained how these differences exactly make OWL more suitable to perform data integration.

A first advantage of OWL is that it has well-founded semantics. This allows for reasoning to be performed. The consistency of the model can be checked to automatically discover inconsistent data. Additionally, classes can be defined as logical constraints, *e.g.* using *intersectionOf* or *unionOf*. The classification of these classes will be performed at runtime. This frees the user from defining the whole classification tree himself, which is often a difficult and error-prone task. It also makes the data integration more future-proof: a reasoner can be used to infer new knowledge from the existing information in the ontology. Applications are able to use this new knowledge. However, inference can be very resource-intensive. Luckily OWL can easily be integrated with various rule platforms to accomplish more resource-intensive reasoning.

A second advantage is the fact that ontologies are more and more being picked up by the industry. A wide range of (mature) tools have been developed to construct an ontology (Protégé (Stanford Center for Biomedical Informatics Research, 2009; Knublauch et al., 2004), Swoop (Kalyanpur et al., 2007; Kalyanpur et al., 2006), *etc.*), visualize it (OWLViz (Horridge et al., 2005; Ellson et al., 2002), Yambalaya (Chisel, 2008; Storey et al., 2002), *etc.*), query it (using SPARQL in for example Protégé or Jena (Carroll et al., 2004)) and reason about it (Pellet (Clark and Parsia, LLC, 2009; Sirin et al., 2007), Racer Pro (Racer Systems GmbH and Co. KG, 2009; Haarslev and Möller, 2003), Drago (Tamilin et al., 2006; Serafini and Tamilin, 2005, *etc.*).

A third advantage is that the main-stream serialization of OWL is XML-based, which allows OWL models to be easily manipulated and exchanged by applications, regardless of platform. Both the domain modelling and the data are sent, so the same model and semantics can be used. Furthermore, it is straightforward to extract the contents of an OWL model into any format. For example, OWL can be serialized into N3 (Berners-Lee, 1998) or Turtle (Beckett and Berners-Lee, 2008), which are more easily read by humans.

The language itself also has some attractive modelling advantages. Ontologies can be divided into different smaller ontologies that inherit from each other, which allows for a modular design and distribution. This is not the spirit of UML. As OWL was designed for distributed information description rather than program definition, it is more intuitive and easier to use for describing real-world concepts than UML. There is a universal class *Thing* that subsumes "everything". This allows for open world reasoning, which states that the truth-value of a statement is unknown until it is explicitly stated to be true or false. It is the opposite of closed world reasoning which states that any statement that is not known to be true is false. For example, a knowledge base contains the statement "Steve speaks French" and the question "Does Steve speak English?" is asked. In a closed world the answer is "No", but in an open world it is "Maybe".

The use of ontologies also has some specific advantages for data integration. The ontology can be used as a global query model or to verify the mappings between the global and the local schemas. These advantages will be further explained in the next section.

4. Existing practical approaches to construct ontologies for data integration

In the previous section, we presented the theoretical foundation of the ontology approach towards information integration. This section builds further on this basis and discusses a number of practical approaches towards actual implementation of the ontology concept in the context of this information integration problem. A lot of tools exist that use ontologies for data integration. They can be compared by (Wache et al., 2001):

- **The use of ontologies**: Ontologies can play different roles in the data integration process. They can be used to make the context explicit and express the semantics or as a global query model.
- **Ontology representation**: The representational capabilities of the used ontologies can vary amongst the different tools.

- **Use of mapping**: Mappings are used to link the ontologies to the actual information and to each other if multiple ontologies are used.
- **Ontology engineering**: The ontologies can be built and re-used in different manners.

These points will be further explained in the following subsections.

4.1. Three approaches for using ontologies

Ontologies can play different roles in the data integration process. Firstly, ontologies can be used to make the context of the data explicit and express the semantics. Three approaches exist: the single ontology approach, the multiple ontology approach and the hybrid approach. All three approaches are graphically presented in Fig. 4.

In the single ontology approach, one global ontology is constructed which expresses the shared semantics between all the data sources for a domain, for example the railway sector. This approach has two disadvantages. First, the single global ontology is very susceptible to changes if more data sources are added to the integration process. Second, it might also be difficult to find the minimal ontological commitment. Each statement in an ontology commits the user of this ontology to a particular view of the domain. If a definition in an ontology is stronger than needed, then we say that the ontology is over-committed. For example, if we state that the name of a person must have a first name and a last name we are introducing a western bias into the ontology and we may not be able to integrate new data using the ontology.

In the multiple ontology approach, each information source has its own ontology. The domain of each data source is modelled in the context of semantics of that data alone. This makes the construction of the ontologies easier, as the minimal ontological commitment problem is less difficult and the ontologies are less susceptible to changes. However, the lack of a common vocabulary makes it difficult to compare the different ontologies. Mappings between ontologies can be defined, but this is a difficult process due to different granularity and aggregation of the ontologies. The problem of semantic heterogeneity has now shifted to the mappings. Two options are available for adding a new data source. A new ontology can be constructed for this data source and new mappings can be defined to all the other ontologies. Or an existing ontology, which already has mappings to the other ontologies, can be re-used to represent this data source. The first option is a lot of work and entails the difficulty of defining the mappings with the other ontologies. The second option is often not possible as the

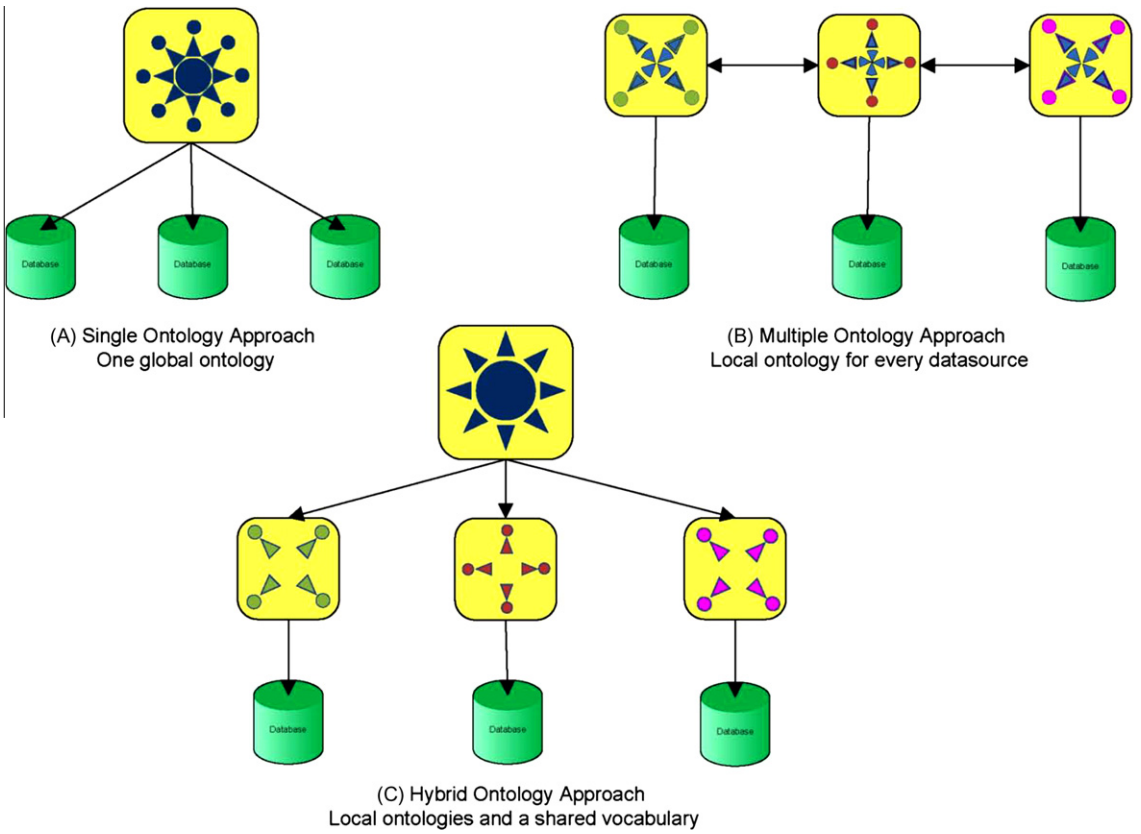

Fig. 4. Three approaches for using ontologies in information systems.

new data source often contains data that cannot be represented in the existing ontology or contradicts with the knowledge present in this ontology. This means that the ontology needs to be adapted. This has to be done without violating the meaning of the data present in the original source for which this ontology was made.

In the hybrid approach, each source has its own ontology and they are built on top of one shared ontology. The shared ontology contains the primitives of a domain. It forms a sort of minimal skeleton of shared knowledge between the different data sources on which the local ontologies can be built. By using the local ontologies, the global ontology is less susceptible to change and the minimal ontological commitment problem is less difficult because knowledge that contradicts each other can be encoded in the local ontologies. For example, if a severe fault of a train is defined differently in different countries, a concept *SevereFault* can be created in the shared ontology. Each local ontology, which represents the data from a certain country, can then contain a concept that inherits from this *SevereFault* concept and encodes the definition that is used to detect a severe fault in that country. Ontologies can easily be compared by using the shared ontology. Each time a new data source is added, a new ontology is constructed extending the shared ontology. This is the approach that was adopted by the InteGRail project.

Secondly, the ontology can also be used as a global query model in the data integration process. The user can specify the query in SPARQL in function of the concepts and relationships in the global ontology. The query is than divided into different sub-queries for the different sources. The results are combined and returned together. This frees the user from specifying the query himself each time for each different data source. The structure of the query model also becomes more intuitive for the user.

4.2. Approaches for ontology representation and mapping

The existing tools for data integration mainly use Description Logic-based languages such as OWL, RDF(S) or OIL. Some tools additionally use Rules to handle the more resource-intensive reasoning tasks. This way a reasoner can be used to check consistency or to infer new knowledge. The remainder of tools mainly use Frame-based representation languages such as Ontolingua. In the InteGRail project all the ontologies were encoded in OWL and SPARQL was used to query these ontologies.

Mappings are used to connect the ontologies to the actual information sources and to each other if multiple ontologies are used. The mappings can be done to a database scheme or directly to single terms used in the database. In the first method, called the structure resemblance method, the ontology is a direct representation of the database schema in a knowledge representation language (one-to-one copy). The second method only defines semantic terms in the ontology that relate to the terms in the database, but does not represent the terms itself in the ontology. The third method, called the structure enrichment method, is the most common approach and combines the two previous methods. A one-to-one copy of the database scheme into an ontology is made. This ontology is enriched with additional terms that express additional domain knowledge about the terms in the database. These additional concepts only relate to concepts in the ontology, *i.e.* no mapping to the original data. This leads to more semantic strength. A fourth and last method simply annotates the source data with semantic information, *e.g.* for example SHOE (Heflin et al., 1999).

The inter-ontology mappings are more difficult to define. Mapping different ontologies is a well-known and much researched topic. This only needs to be done in the multiple ontologies approach, see Section 4.1. The mappings can be defined by the users or domain experts. This allows for much flexibility, but is a tiresome and error-prone task. Users can define arbitrary mappings, even if they don't make sense or produce conflicts, which may corrupt the semantics of the ontology. One can also define intuitive or well-founded semantics for the mappings between different concepts of the ontology. Methods are still under development to do this in an automated manner.

The exploitation of a robust and scalable service platform to implement one of the mapping methodologies described above is of paramount importance to transfer the conceptual data integration ideas into an industrial usage. In other domains, such as telecom and complimentary to the previous described examples, commercial platforms for this structured and semantic-enabled data integration are emerging (Ontology Systems, 2009a,b; LogicU, 2008; Metatomix Inc, 2009a,b).

An open-source variant, based on the same principle, *i.e.* exposing the information contained in existing systems in an ontology-based manner can be found in (Bizer & Cyganiak, 2006). It is an implementation of the third approach as described earlier in this section. D2R serves as a virtual triple store layer on top of a relational database. A triple store is the common name used to refer to ontology-enabled databases. After all, an ontology is represented as a graph-like model, consisting of a subject, predicate and object. These three items together form a triple. The D2R library is able to publish the contents of a relational database as a virtual RDF-Graph on which SPARQL-queries can be executed. This library then performs an on-the-fly conversion between the SPARQL-query and corresponding SQL-queries. These retrieve the needed information from the relational database. Consequently the result-sets are converted into semantic RDF data, and are then returned as an answer on the original SPARQL-query. One of the advantages of this approach is the online conversion of the SPARQL-query into SQL-queries. The results returned are therefore always an online and up-to-date RDF-representation of the tables and rows in the relational database. Therefore, extra care must not be taken in order to keep the triple store synchronized with the relational database. Additionally, extra semantic information can be added to the transformation. These extra annotations can be statically defined in the mapping files describing the relationships between the tables and columns of the relational database and the concepts and relationships in the ontology. In a first implementation phase this D2R Model could be used as temporary solution, and later on a transition could be made towards a real semantic triple store without the need to change the logic of the application for a second time.

4.3. Approaches for the construction of an ontology

Well-known and good methodologies exist to construct an ontology, for example TOVE (Gruninger and Fox, 1995), ENTERPRISE (Uschold and King, 1995) and METHONTOLOGY (Fernandez et al., 1997). There are five widely accepted stages for building an ontology (Pinto and Martins, 2004):

- **Specification**: The purpose and the scope of the ontology are identified.
- **Conceptualization**: A conceptual model of the ontology is constructed. It consists of the different concepts, relations and properties that can occur in the domain.
- **Formalization**: The conceptual model is translated into a formal model for example by adding axioms that restrict the possible interpretations of the model.
- **Implementation**: The formal model is implemented in a knowledge representation language, for example OWL.
- **Maintenance**: The implemented ontology has to be constantly evaluated, updated and corrected. To update an ontology, the previous steps can be used.

Additionally there are three activities that should be done during all the stages, namely knowledge acquisition (for example brain storming or questionnaires for domain experts), documentation (for example what was done and how) and evaluation of the quality of the ontology. The three mentioned methodologies only differ in the way these stages or activities are filled in. These approaches do not give the domain experts and users active roles in the development process. They are only consulted during the specification stage, while the conceptualization, formalization and implementation stages are mostly covered by the ontology engineer. The domain experts and users are often also consulted during the evaluation of the ontology.

As a result, new methodologies have been developed which involve the users and domain experts more in the ontology development cycle, namely HCOME (Kotis and Vouros, 2006) and DILIGENT (Vrandevic et al., 2005). Both approaches emphasize that engineers must minimize their involvement during the engineering process of the ontologies, giving more control to the domain experts. As a consequence, both methodologies also consider distributed settings and evolving ontologies and elaborate on issues such as collaboration, version management and merger of ontologies. Both approaches allow users to develop individual conceptualizations of their domain and share them with other users. These conceptualizations are then merged to achieve a global conceptualization of the domain. The main difference between both approaches is that DILIGENT employs a "control board" that guides and oversees the ontology development cycle. Many data integration tools encourage the use of these methodologies. The reuse of ontologies is also encouraged as this makes the application interoperable with other applications that use this ontology.

5. The Network Statement Checker: results

The ontology-based approach towards data sharing and integration, as presented in the previous sections of this paper, will be illustrated in this section by means of an example use case implementation, namely the Network Statement Checker (InteGRail Consortium, 2008).

European legislation to enhance the open market and competition in the railway has split railway undertakings into operators and infrastructure managers. The needed technical information to verify the feasibility of running a specific train on a specific track is in the so-called network statement. One can define a strict syntactic protocol to specify this network statement, for example the position of the track, the electrification properties, such as voltage, maximum current, installed safety system, *etc*. This allows an automation of the train-track matching in a tool called the Network Statement Checker (InteGRail Consortium, 2008). The need for a semantic level becomes apparent when one wants to integrate new datasets, for instance the network statement of a second country to organize a new international train service. Voltage and current are rather straightforward to integrate; safety systems on the other hand are more problematic: several names exist for similar systems (for example the Belgium LAT or crocodile system). Sometimes safety systems are identical (synonyms), sometimes one system is a superset of the other.

Another semantic issue is the presence of duplicate records in a newly integrated dataset, for instance the border stations on a rail network. For the purpose of finding connecting track segments, border stations are to be treated as a single data record. For a passenger information system or a ticketing system, the two instances will be considered as belonging to a different country, with possibly a different language for the public address system or a different tariff plan. Therefore two instances are kept and the semantic relation that they are the same physical location is added to the information system. We can note here that a balance has to be found between merging the individuals together – *i.e*. giving them the same type – or keeping them separate and matching them onto each other to reflect their relationship. Therefore, the modeller has to make a decision between realizing the individuals on the same concept or creating multiple concepts for different meanings for those individuals.

The border station example already shows that one concept (*station*) is given different meanings by the different railway stakeholders, hence the need to store with the data also its meaning. This semantic information has to be added by domain experts. For such a vast industry as the railway industry, thousands of companies active in Europe, the largest

ones, DB and SNCF having more than 200,000 employees, the management of semantic information is necessarily a distributed effort.

The Network Statement Checker used by an operator is based on an infrastructure database describing the rail network. A passenger information system providing vocal and visual travel information, on-board the train and at stations also needs a description of the rail network. In these cases, *voltage* and *safety-system* are not relevant but new concepts as *language area* or *connecting trains* are introduced. Overlapping concepts such as *GPS position* are the most important towards information integration. For the purpose of the Network Statement Checker, this will be used to identify concatenated track segments, which means that consistency of the positions is more important than accuracy. For an on-board passenger information system, the GPS position is needed for automated announcements of the current and next station. Accuracy is of more importance here, certainly if we want the on-board system to detect and announce a platform alteration when the train is entering the station. A semantic system that allows an elaborated and extendable concept of *location information* is therefore needed.

Many efficiency improvements in the railway industry are dependent on information integration, for example optimal route planning in terms of cost or speed via a Network Statement Checker, condition-based maintenance via track-wheel mutual measurements, improved maintenance scheduling with evaluation on the impact on rail traffic or improved service to passengers through real-time update of passenger information, including schedule changes due to immediate maintenance actions.

It is clear that with more and more cross-border traffic, information from different systems in different countries need to be integrated and converted in order to guarantee that the information from all countries can be taken into account by the application. Therefore, it is paramount that a commonly agreed data model is specified. We claim that an ontology-based methodology is highly suitable to do this. The Network Statement Checker application integrates information of The Netherlands and Belgium. The heterogeneity of the data is not only to be found in the structure of that information, but also in the information systems that contain this data, being in a text book spreadsheet style and in a relational database.

The use of ontologies as a self-descriptive and extendible mechanism can hide this heterogeneity from the application. Existing libraries can be used to convert on-the-fly the data from the legacy systems into individuals of the commonly agreed ontology. Additionally, given the current development of those libraries and the emergence of commercial platforms to support this integration, it can be expected that ease of use and performance will improve significantly.

Reasoning can support intelligent abstraction of the domain logic from the application using the information from different systems, as is presented in this paper, without the need to implement this logic in the application itself. Additionally, transmission of data between these systems is supported by means of standardized serialization formats, *e.g.* based on XML in RDF/XML.

It thus allows decision makers to make better decisions once they have the right information at hand about their own processes and those of their partners. More concrete, the Network Statement Checker integrates the network statements of difference countries in a single semantically enabled information model. A screenshot of the application is illustrated in Fig. 5. The main panel contains a map of, in this case, The Netherlands and Belgium, displaying the Dutch and Belgian railway network. The top-right hand side contains an overview of the available trains. On the middle-right hand side of the screen, information can be obtained on the selected train. Some of these characteristics are voltage, maximum allowed axle load, *etc*. The lower-right hand side allows selecting a starting node and an ending node between which a route should be calculated, given the selected characteristics of the train. Optionally, specific characteristics to be considered can be enabled or disabled as well. The resulting list of connecting track sections is displayed in the lower panel of the screen. The route with all possible and prohibited diversions is also graphically displayed on the map.

It is a web-based application that allows online access to the network statements of national infrastructure managers. The user of the tool can select a route on the European railway map and can find information about the characteristics of each track section on a route. This tool provides information needed to determine whether a route can be used, from a compatibility point of view, for a new future railway service an operator intends to offer to his customer.

Additionally, the user can select a start and end node between which the route is to be checked. The Network Statement Checker application then calculates a route between those two points, using the information converted in the agreed ontology format, as retrieved from the heterogeneous legacy systems integrated in the platform, and matches it with the required characteristics as specified by the user. A proven algorithm, *e.g.* Dijkstra shortest path, is used to calculate the actual path using the ontology A-Box as dataset.

In the present railway situation, the information this tool deals with can be found in each country's individual network statement. It is the obligation of each infrastructure manager to publish this document annually. The network statements of each country already have the same structure, but the information itself is not provided according to any standard. This makes retrieval and analysis of the network information an error-prone, difficult and time-consuming manual task. The Network Statement Checker will reduce the time and effort needed to retrieve the information, by automating this procedure. This automation is facilitated by the ontology-based approach. The national network statements of Belgium and The Netherlands have been integrated, either by converting the information available in text-documents into an ontology format, or by defining an automated mapping, using D2R (Bizer and Cyganiak, 2006), on a relational database already containing a digital version of the network statement.

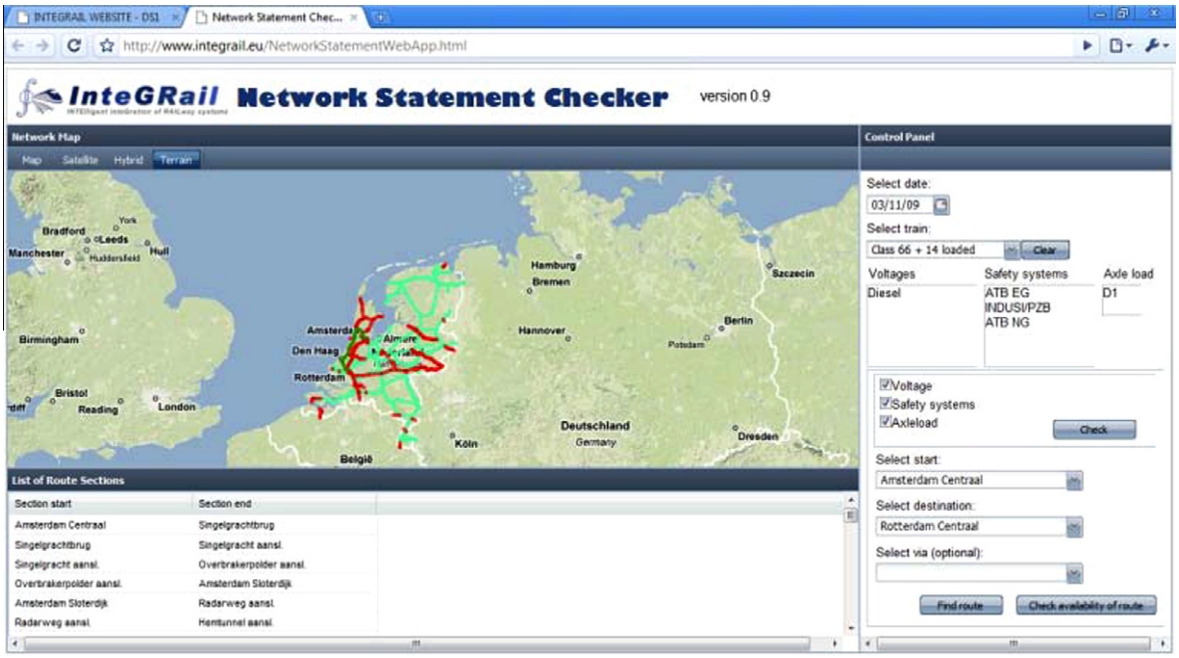

Fig. 5. Screenshot of the Network Statement Checker web application.

5.1. The Network Statement Checker ontology engineering process

In Section 4.3 we presented a number of ontology engineering processes. As indicated by Pinto & Martins, there are five widely accepted stages for building an ontology (Pinto and Martins, 2004). These are in order of execution: Specification, Conceptualization, Formalization, Implementation and Maintenance. The steps in the METHONTOLOGY (Fernandez et al., 1997) methodology coincide largely with these. We have adopted this approach in the engineering process for the Network Statement Checker ontology. Alternative methodologies, which have a more elaborate and extensive involvement of the users, such as Diligent or HCome, were considered not be necessary because of the strict regulatory nature of the railway domain and the fact that the knowledge about the domain was already largely present due to the involvement of the project partners.

Together with partners from the industry, co-authors of this paper, the scope of the envisaged application was set out. As indicated in the introduction of this section, the goal was to develop a transparent application to present and integrate the information contained in the network statements of the railways in Belgium and The Netherlands. This application should also be easily extendible with potential new network statements from other countries. Additionally, a route planning functionality was foreseen, so that the requirements of the rolling stock and the characteristics of the track were matched and a potential route between the two destinations is calculated.

Once this application specification was finalized by the domain experts, a meeting was organized with the modelling engineers from Ghent University – IBBT and the domain experts from DeltaRail and Televic, in order to define the conceptual domain model. The resulting model is described in Section 5.2. Given the complex nature of the domain, the input from the industrial partners was of extreme importance to align the perception and meaning of the included concepts. During this meeting the two different views on the domain were defined, as can be seen in Figs. 6 and 7. Additionally, an abstraction of complex stations was agreed and was named the NetworkNodeIOPoint. This particular concept is illustrated in Fig. 8. All information and discussions resulting from the brainstorms during this 1-day meeting were written down in textual descriptions as well as graphical diagrams. For a more elaborate description of the difference between these concepts, we refer to Section 5.3.

With this conceptual agreement in place, the ontology development started. The first version of the ontology model was created without the involvement of the domain experts, thus solely by the modelling engineers. Starting from the hybrid approach, using a core ontology with a core vocabulary and domain extensions, as developed in InteGRail (InteGRail Consortium, 2009), the correct terms agreed by the wider industrial partners in the InteGRail Consortium were pinpointed. The development of the core ontology by the InteGRail Consortium proved to be a major task. Therefore, to reach a consensus, no specific logic constraints were specified in this core model. It should therefore be seen as a common vocabulary agreed by a wide range of railway stakeholders. From these terms, more specialized terms and additional concepts and relationships were defined in the Network Statement Checker ontology. As such, this ontology was constructed as a domain extension on the agreed core ontology. Once this first version of the ontology was drafted, a second meeting with the domain experts from

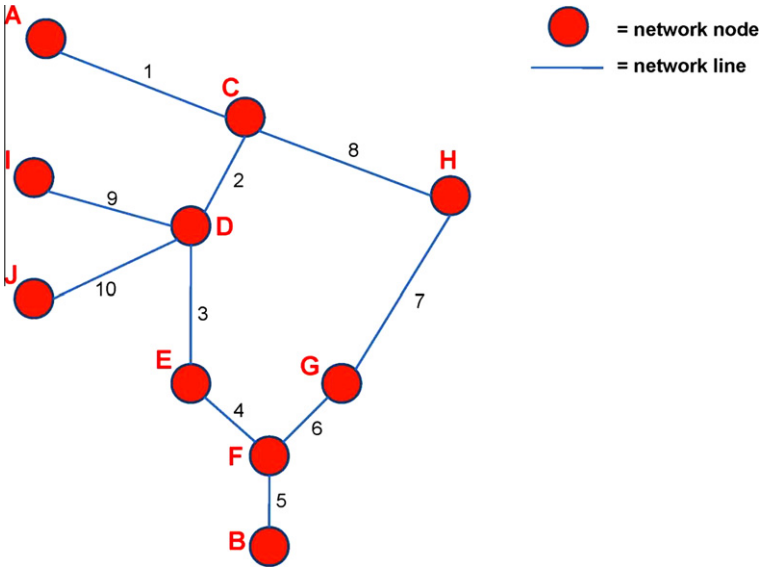

Fig. 6. Logical representation of nodes and edges in a railway network.

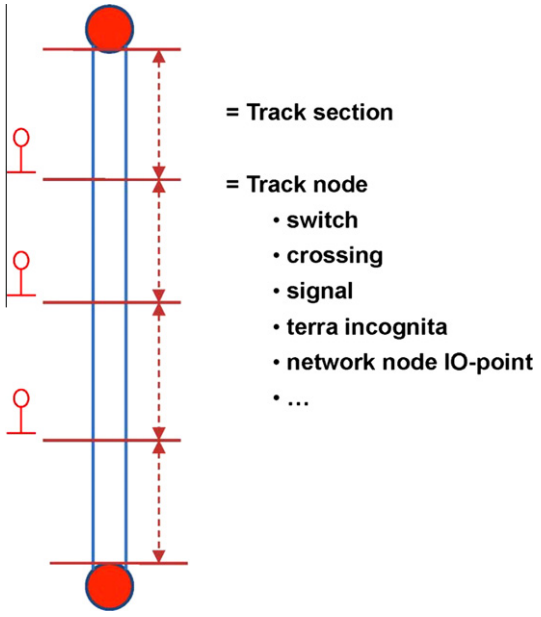

Fig. 7. Decomposition of a logical network line into physical track sections and track nodes.

DeltaRail and Televic was planned to validate and finalize this ontology formalization. Additionally, the necessary queries were defined. These queries were specified according to the agreed Network Statement Checker ontology, and are as such independent of the underlying legacy information systems used in both countries. However, dual system specific queries were internally specified in order to be able to verify the results of the ontology queries.

With the Network Statement Checker ontology in place, the independent implementation of both approaches was started. Close cooperation between the modelling engineers as well as the domain experts was pursued in this phase. In Belgium, a D2R (Bizer and Cyganiak, 2006) conversion specification was defined between the existing tables in the relational database and the concepts in the ontology. This was configured and deployed on a machine at Televic, Izegem, B. In parallel, a different implementation approach was adopted for The Netherlands. The Dutch network statement was only available in textbook format. Therefore, we adopted a native ontology approach, implemented in Java by using the established Jena toolkit (Carroll et al., 2004). The information was stored on a machine at the DeltaRail offices in Utrecht, NL and made available for integration. With this deployment in place, the results of the SPARQL ontology queries were verified with the results obtained from

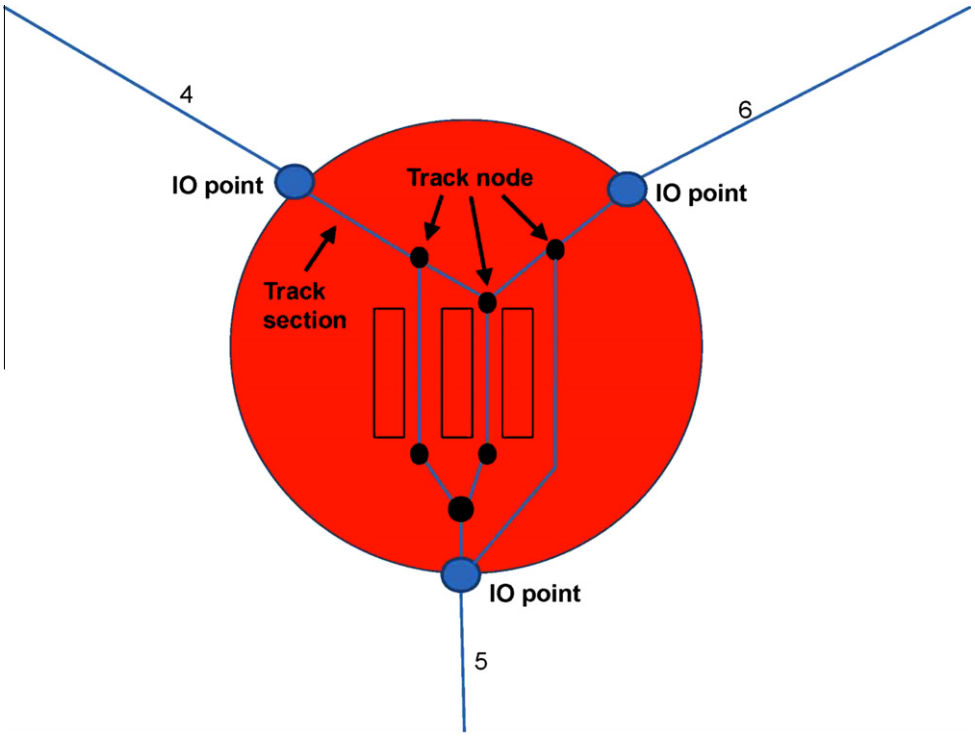

Fig. 8. Abstraction of the detailed Physical Track Node.

the native legacy systems. Once this verification was completed, both systems were integrated and used by the Network Statement Checker front-end web-based application. Finally, the merged results were verified with the results obtained directly from the legacy systems. The input from the domain experts of DeltaRail and Televic towards this validation was of paramount importance as well as their support in the demonstration as part of the InteGRail stand at the InnoTrans fair 2008, Berlin, Germany.

5.2. The Network Statement Checker domain model

In Section 6, we elaborate on the concept of a core ontology, with domain extension ontologies as a means of structuring the engineering process, named the hybrid ontology approach. This process also promotes the reuse and adaptation of existing models. In order to be compliant with this engineering process, the Network Statement Checker ontology has been constructed as an extension of the InteGRail Core Ontology, as can be found in (InteGRail Consortium, 2009).

Two major requirements should be distinguished for this ontology. On the one hand, there is the need to have a model for the different items required to describe a railway network. On the other hand, the corresponding characteristics of those items must be modelled as well. Furthermore, we have taken into account a distinction between the logical and the physical railway network. The concept of a logical network is graphically presented in Fig. 6.

In this illustration, the items 1–10 refer to the line segments, the *Network Lines*, and the items A–J denote the junctions of two or more of those line segments, the *Network Nodes*. A network line is the concatenation of railway track sections that have the same characteristics, in the context of the network statement. This is illustrated in Fig. 7, where four physical track sections make up a single logical network line. For the Network Statement Checker, it is sufficient to work on the logical level. After all, if the characteristics of all track sections within the network line are the same, there is no need for a further decomposition to know whether a certain route between two nodes is available for given rolling stock.

Another abstraction introduced in this ontology, is that of a *NetworkNodeIOPoint*. According to the same principles and following the same arguments, the detailed physical modelling of a junction is not taken into account. If certain network characteristics exist on an incoming network line and the junction is to be used as a transit node, there should be an outgoing network line with the same characteristics; otherwise there is no through-route for rolling stock which such characteristics through this junction and then this junction is an end-point in the network. This abstraction is presented in Fig. 8.

5.3. The Network Statement Checker ontology

Having introduced the general concepts of the domain model in the previous section, Section 5.2, this section presents the ontology model as implementation of that conceptual model. Two aspects of the ontology are detailed. Firstly, Section 5.3.1

presents the static ontology giving the semantic representation of the domain model described in Section 5.2. Secondly, the dynamic extension of the model is described. This dynamic extension facilitates a more intelligent view on the available information, which can additionally be augmented with information coming from other systems, *e.g.* such as a track monitoring application, so that a filtered dataset can be provided as well to the, by the Network Statement Checker application, implemented route planning algorithm. This is detailed in Section 5.3.2.

5.3.1. Static ontology model

After considering the InteGRail Core Ontology, only a limited number of extra concepts had to be introduced in the Network Statement Checker ontology. More extensions were required however, to include the necessary relationships in the ontology to link the concepts together and to create a well-designed and fundamentally complete model. To ease the description of the concepts and relationships in the Network Statement Checker ontology, the major items are illustrated in Fig. 9.

At the root of the concept-tree an abstract class is defined, namely the *NetworkTopologyElement*. This is the main concept from which all other Network Statement Checker concepts are derived. A distinction is then made between edges and nodes. After all, as has been described in the previous section, the network is modelled in a graph-like structure. Each of these edges and nodes in their turn has a logical and physical representation. The logical representation is given by the *LineNetworkEdge* and *-Node* concepts. The instances of these concepts represent the ontology equivalent of the concepts defined in Fig. 6. On the other hand, the physical concepts are defined by the *TrackNetworkEdge* and *-Node* concepts, representing the items as presented in Fig. 7. Further subclassing of the nodes is rather straightforward and is represented in Fig. 9.

The physical *TrackNetworkEdges* are linked to the logical *LineNetworkEdges* by means of the *isComposedOf* relationship. After all, the physical characteristics, as defined by the concept *TrackCharacteristic*, are linked to the physical *TrackNetworkEdges*. In order to be able to take these characteristics into account for the representation of routes and to be able to check whether a certain type of train is allowed to use a particular given route, the *LineNetworkEdges* should be able to refer to these characteristics as well. To create a full graph-like model for the complete railway network, the edges are linked to the nodes by means of two relationships, namely *startsAt* and *endsAt*.

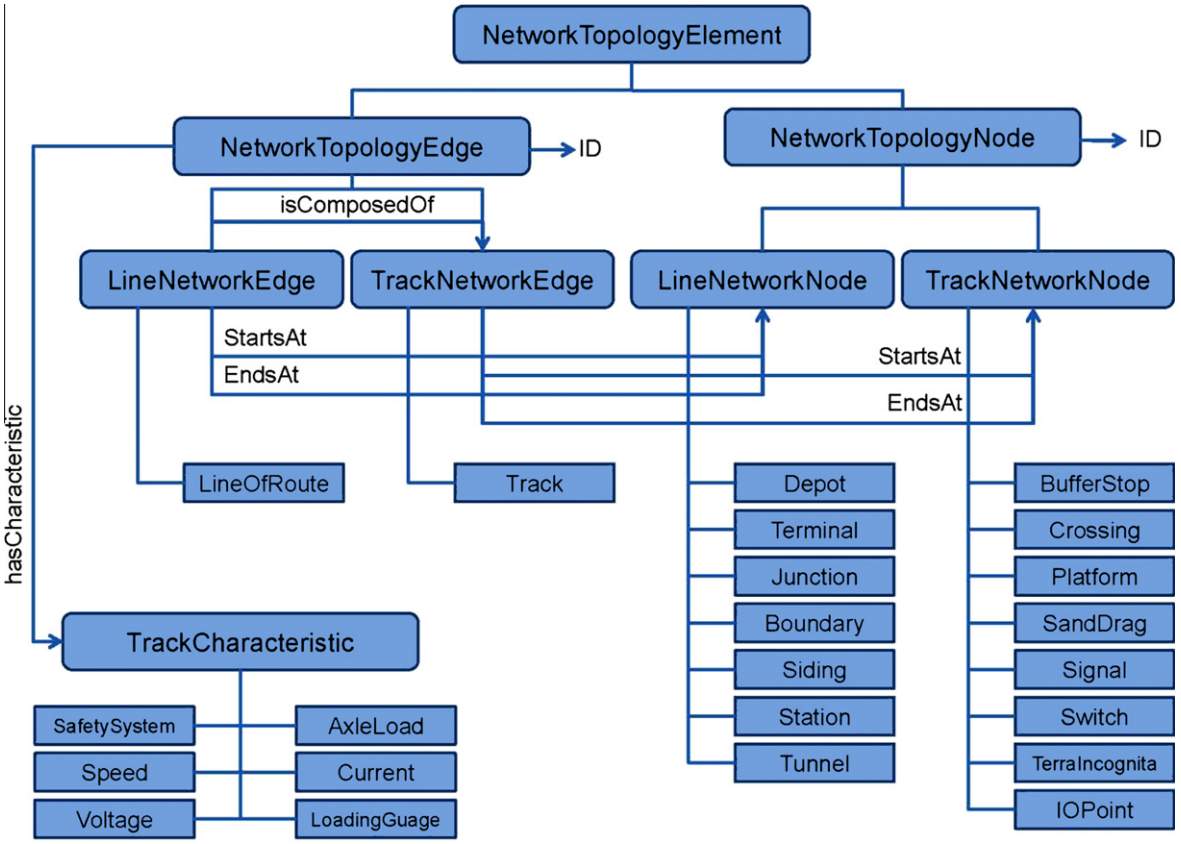

Fig. 9. The Network Statement Checker ontology.

5.3.2. Network Statement Checker ontology extended with intelligent filtering

The previous section detailed the semantic representation of the domain model for the Network Statement Checker application. However, this model does not exploit the ontology methodology to its full extent because constraints were not used in this first model.

Therefore, by making use of the inheritance mechanisms provided by OWL, the ontology has been extended with a number of additional concepts, defined by means of constraints, as subconcepts of the *TrackNetworkEdge* and *TrackNetworkNode*. A first potential extension is to specify predefined *TrackNetworkEdges* for given train configurations. *E.g.* certain *TrackNetworkEdges* might not be suited for usage by "Class 66" train configuration. Instead of having these constraints checked by the applications, additional concepts are introduced in yet another extension of the Network Statement Checker ontology, specifying by means of description logics axioms, the facts that have to be true for a given *TrackNetworkEdge* to be classified as *Class66CapableTrackNetworkEdge*. As a consequence the axioms by which this capability is specified can be consistently refined by each and every local ontology integrated in the application platform. Only the constraints in the ontology deployed on top of each local system should be altered in order to reflect the rules and regulations of this county. Given the nature of the partitioned deployment, *i.e.* for every single country or distinct dataset a separate server, potential contradicting constraints on the same data are avoided. After all, there is only one logical place in the platform, where certain data is exposed. We do not, however, take load-balancing or redundancy techniques into account, since we are convinced that in such situations, these implemented techniques fall on the responsibilities of the same administrators, thus minimizing the potential for conflicts. Additionally, this specification can be altered over time as the rules and regulations change. The following definitions give a few examples illustrating the different restrictions for the same concept in The Netherlands compared to Belgium.

In The Netherlands, for a given *TrackNetworkEdge* to be denoted as Class 66 capable – ***NL****Class66CapableTrackNetworkEdge* - at least the following statements have to be true:

- **subClassOf** nso:*Class66CapableTrackNetworkEdge*
- *core:hasTrackCharacteristic* **min 1** nsoNL:*ATB NG*
- *core:hasTrackCharacteristic* **min 1** (*nso:AAxleLoad or nso:D3AxleLoad*)

These constraints specify that at least the *ATB New Generation* safety system should be installed on the track and the train, and that either *A Axle Load* or *D3 Axle Load* is allowed on this specific track. These concepts are defined in the local extension of the Network Statement Checker ontology (indicated by the *nso**NL*** namespace) as subconcepts of the *Class66CapableTrackNetworkEdge* concept, which is defined as a term in the common Network Statement Checker extension of the core ontology, illustrated in Fig. 9. For the Belgian ontology specification, the constraints should be changed, and in particular the safety system restriction. This means that the service exposing the Belgian ontology-based network statement (indicated by the nso**B** namespace) should have a definition of ***B****Class66CapableTrackNetworkEdge* as follows, specifying the *Crocodile* safety system as compulsory:

- **subClassOf** nso:*Class66CapableTrackNetworkEdge*
- *core:hasTrackCharacteristic* **min 1** nsoB:*Crocodile*
- *core:hasTrackCharacteristic* **min 1** (*nso:AAxleLoad or nso:D3AxleLoad*)

Using this pre-processing of information, the Network Statement Checker application should not be adapted for every change in rules and regulations and additionally should not know the specifications of every country being queried. After all, the application only queries the services for *Class66CapableTrackNetworkEdges*, exploiting the inheritance mechanism. This reduces also the dataset being used by the Dijkstra algorithm in the route planning phase, as only the suited edges are being taken into account by the algorithm.

Because individuals can be realized as belonging to more than one concept, further constraints can be added to the specification, *e.g.* by integrating information coming from other systems, such as track monitoring- or disturbances information systems. Exploiting the fact that an individual in the ontology is uniquely identified by its URI, this URI can be used by multiple information sources to link information denoting the same domain object. Therefore, the track monitoring information source uses the same URIs as those used in the Network Statement Checker to attach additional information to those representations, such as *Statuses, Observations, Faults*, *etc*.

Using this extra information, the modelling of *UnavailableTrackNetworkEdges* becomes as follows:

- **subClassOf** *TrackNetworkEdge*
- (*hasStatus* **some** *NotOKStatus*) or (*hasObservation* **some** (*hasFault* **some** *SevereFault*))

The above statement specifies that for a *TrackNetworkEdge* to be classified as unavailable it should of course be a *TrackNetworkEdge* and it should either have a status which is *NotOK* or it should have an *Observation* which leads to a *SevereFault*. It is clear that more fine-grained definitions could be specified should the need exist to distinguish between more than a binary situation. Because of the Open World Assumption paradigm adopted by the reasoning mechanisms, the set of *UnavailableTrackNetworkEdges* would have to be subtracted from the complete set with all *TrackNetworkEdges*. This in fact boils

down to closing the world. It is after all not possible to draw conclusions and to classify concepts or realize individuals in the situations, where no information is available.

Additionally, context information not directly related to the classification of the *TrackNetworkEdges* can be included as well. By re-using the URI of the *TrackNetworkEdge, Status, Observation* of *Fault*, this information can be attached to the individual concerned. Some of this information could be the date and time of the *Observation*, the weather condition at the moment the *Fault* was detected, *etc.* Fig. 10 illustrates the enhanced ontology model.

5.4. Deployment details

As has been detailed previously, we have adopted the hybrid ontology approach towards information integration. This inherently facilitates a distributed approach, which is presented in more detail in this subsection.

The core ontology, as agreed between the partners in the InteGRail consortium (InteGRail Consortium, 2009), should be considered as a common vocabulary. This ontology is very stringent and the semantics cannot be changed and should be adhered to by all the partners using or extending this core vocabulary. Therefore, every participating distributed information system should expose its data in accordance with at least this commonly agreed core ontology. Of course it can define extensions, *i.e.* subclasses, in its locally deployed ontology T-Box. As a consequence, to include the local rules in the deployment, additional constraints should be specified in the extended ontology.

The queries to retrieve the information from these distributed systems are – and should only be – defined according to the terms in the common core ontology. After all, these terms are the common denominator amongst all information sources participating in the distributed ontology-based information integration platform. Incidentally, this means that the core ontology can be seen as the mapping mechanism between the information in distinct sources in the platform. Using the example from Section 5.3, the concept *Class66CapableTrackNetworkEdge*, is known by all systems exposing the network statements in an intelligent manner. However, the different constraints are modelled locally and specifically according to the local rules and regulations. Therefore, it is up to the local administrator to define the exact constraints. There is no additional mapping language or mechanism specified. As long as the terms of the core ontology are used correctly, their semantics are inherited automatically. An exact and common definition of this core ontology is of extreme importance in order to avoid any ambiguity. However, we are convinced that in the railway domain, with its strict nature, sufficient regulatory organisations exist to monitor and enforce the adherence to these specific constraints.

As a result of this architecture and deployment approach, only the appropriate information is retrieved from the distributed information systems. The information integration, *i.e.* the merger of the individual results, is performed after the

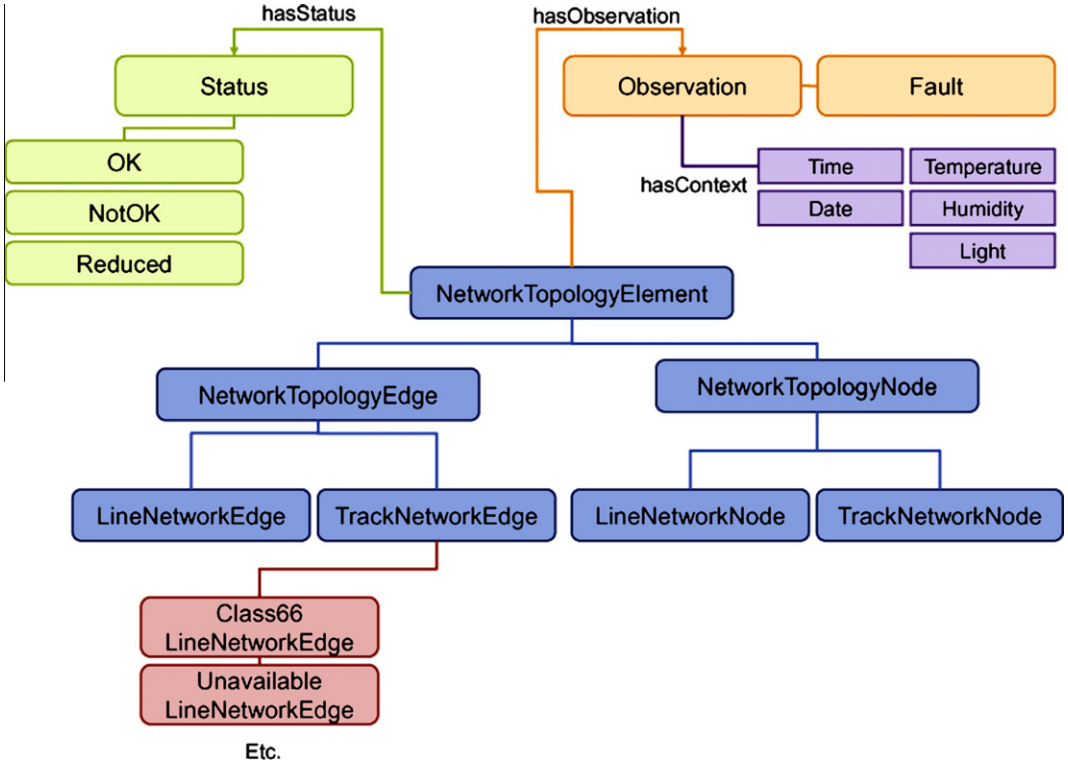

Fig. 10. Expanding the Network Statement Checker ontology with context information and observations coming from other monitoring systems.

distributed execution of the queries. These queries, and thus the reasoning by means of classification and realization which is triggered by the query invocation mechanism, are executed locally and only the results of these queries, reflecting the local knowledge and implementation of the generic common vocabulary term, are returned to the front-end application. The deployment is graphically illustrated in Fig. 11. The common core vocabulary, denoted with the namespace prefix nso, is extended with 2 distinct ontologies, as described earlier in a local manner. One of the extensions represents the Belgium network, with nso**B** namespace prefix, and one represents The Netherlands, with nso**NL** prefix. The two ontologies are deployed on individual machines in the two countries. The A-Box of these individual ontologies is populated with the local information of those two network statements.

5.5. Back-end implementation for the Network Statement Checker

As indicated in the previous sections, the network statements of both the Belgian and the Dutch railway networks are contained in two totally different legacy systems, using two different persistency formats. On the one hand, the information was available in a relational database for the Belgium network statement. On the other hand, only textbook information was available for The Netherlands. Therefore a naive common back-end implementation was insufficient. However, using the ontology approach described in this paper, a single implementation of the domain logic was achieved. Instead, an additional conversion step was added to the overall software implementation. This part enables the legacy system specific format to be converted into the common agreed ontology format. As such, the Network Statement Checker application could integrate the information from both countries in a transparent manner. However, the two systems are still owned and maintained by different stakeholders. By no means should the information be stored in a single information system, because this information is owned by different stakeholders and it is desired that the current legacy systems remain operational. The introduction of conversion tools, such as RDF123 (Han et al., 2008) and D2R (Bizer and Cyganiak, 2006), ensures that the ontology representation of that data is always coherent with the original data. Additionally, in the case a new system is to be developed by a stakeholder, the adoption of a native ontology-based implementation, replacing the older legacy system, does not influence the logic of the integrating application. This has been demonstrated by the inclusion of a Jena (Carroll et al., 2004) based information source in the integrating Network Statement Checker application presented in this use case. Graphically, this is illustrated in Fig. 12.

As can be seen from Fig. 12, starting from the bottom of the diagram, a number of legacy systems can be found, which provide the data in their own legacy format. In the current use case this is either stored in a relational database, or written down in textbook format. The conversion modules included additionally in this platform, either convert the contents of the

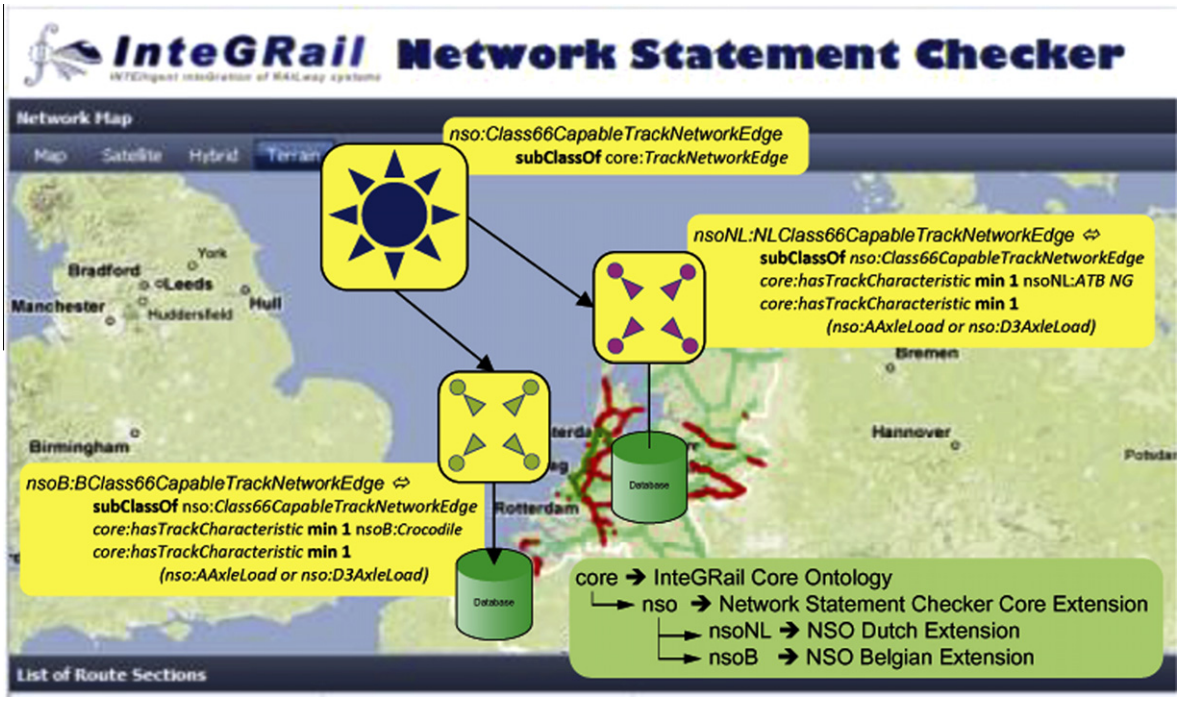

Fig. 11. Distributed deployment illustrating the hybrid ontology approach.

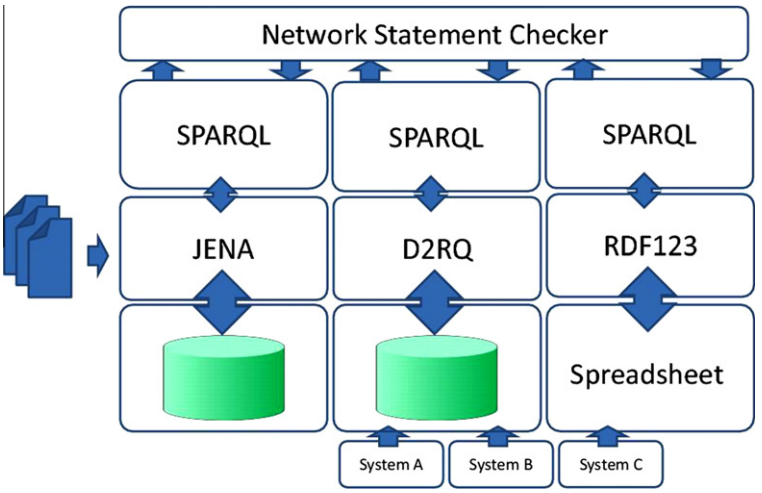

Fig. 12. Back-end platform for the Network Statement Checker application, illustrating how the heterogeneous information from both countries is transformed into a common agreed ontology format.

tables of the relational database into instances of the concepts in the ontology or read in the textbook files and use the native Jena library to convert it into the ontology format and store it using a specific ontology-enabled persistency mechanism. In this case a database was also used to store this converted information, but the organization and contents of this database is entirely managed by the Jena library. Both these approaches support a SPARQL interface. Using these SPARQL interfaces, the Network Statement Checker application can issue the same queries, as presented in Section 5.6 to both deployments without any adaptation.

Illustrated in the third column is another example of the integration of information in an additional approach, when the information is available in spreadsheets. Again, a mapping can be defined between the cells of the spreadsheet and the concepts and relationships of the ontology. As a result, the same SPARQL-queries can then transparently be issued to this deployment of yet another legacy system which stores its information in its own legacy format.

5.6. Performance evaluation of the Network Statement Checker

We have evaluated the ontology-based approach by means of an example integration application as described in the previous section. The application uses the Network Statement Checker ontology as a means of information modelling and integration. The network statements of both The Netherlands as well as Belgium were integrated in this way. For the Dutch network statement, the information was available in textbook format and had to be converted into the ontology format. This was a one-time conversion process, and the Jena (Carroll et al., 2004) library with a MySQL relational database persistence layer was used for this purpose. The Belgian network statement was available in a relational database. In order to integrate this information into the same ontological model D2R (Bizer and Cyganiak, 2006) was used to create a mapping between the tables of the relational database and the concepts and the relationship of the ontology. Example mappings between the tables of the relational database and the concepts of the ontology are given in the following paragraphs.

The *LineNetworkNode* instances of the ontology are created from the stations in the relational database. As can be seen, a new instance of *Station* is created for every row in the table *tbl_station*. Since the ontology model uses inheritance to define that every *Station* is also a *LineNetworkNode*, the *Station* individuals are also returned when the Network Statement Checker queries for *LineNetworkNodes*.

```
map:tbl_station a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern ‘‘tbl_station/@@tbl_station.stationID@@’’;
  d2rq:class core:Station;
  .
```

Additional properties were defined according to the ontology model to store information of the characteristics of those nodes, *e.g.* concerning their location or name. Example mappings for the *latitude, longitude* and *hasName* properties as specified in the core (geo namespace prefix) and Network Statement Checker ontology (nso namespace prefix) model are given below:

```
map:tbl_station_GPSPosX a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_station;
  d2rq:property geo:latitude;
  d2rq:column ''tbl_station.GPSPosX'';
  d2rq:datatype xsd:decimal;
  .
map:tbl_station_GPSPosY a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_station;
  d2rq:property geo:longitude;
  d2rq:column ''tbl_station.GPSPosY'';
  d2rq:datatype xsd:decimal;
  .
map:tbl_station_shortName a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_station;
  d2rq:property nso:hasName;
  d2rq:column ''tbl_station.shortName'';
  .
```

The first definition creates instances of *LineNetworkEdge* for every row in the table *tbl_net*. Using the foreign keys for the start- and end-points of these lines, a mapping is defined between these keys and the *startsAt* and *endsAt* relationships defined in the ontology. This is illustrated in definitions 2 and 3. The mappings for the line characteristics are also specified. An example for the *axle load* and *voltage system* is given in definitions 4 and 5.

```
(1) map:tbl_net a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern ''tbl_net/@@tbl_net.netID@@'';
  d2rq:class core:LineNetworkEdge;
(2) map:tbl_net_startStationID a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_net;
  d2rq:property core:startsAt;
  d2rq:refersToClassMap map:tbl_station;
  d2rq:join ''tbl_net.startStationID = tbl_station.stationID'';
  .
(3) map:tbl_net_endStationID a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_net;
  d2rq:property core:endsAt;
  d2rq:refersToClassMap map:tbl_station;
  d2rq:join ''tbl_net.endStationID = tbl_station.stationID'';
  .
(4) map:tbl_net_axleLoad a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_net;
  d2rq:property nso:hasAxleLoad;
  d2rq:column ''tbl_net.axleLoad'';
  .
(5) map:tbl_net_voltage a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:tbl_net;
  d2rq:property nso:hasVoltage;
  d2rq:column ''tbl_net.voltage'';
  .
```

For a detailed presentation of the constructs in the D2R mapping language, we refer to the excellent description given by Bizer and Cyganiak (2006).

A number of queries were defined in order to retrieve the information from the ontology-based data-stores. The queries are all formulated using the SPARQL (Prud'hommeaux and Seaborne, 2008) query language. SPARQL is commonly used to query ontology models, mainly RDF-based. However, since the queries are mainly A-Box queries for individuals of a given concept defined by means of constraints, SPARQL is sufficient. It uses a triple pattern matching mechanism to find the triples in the dataset of the ontology model which satisfy the fixed concepts defined in the query. The results in those matches for the unspecified concepts are then bound onto the variables in the query. Query 1 retrieves all *LineNetworkEdges* from the

repository, while query 2 retrieves all the nodes interconnecting these *LineNetworkEdges*. The last query retrieves all characteristics of the track sections. The actual queries are presented in the Table 1.

The execution time of these queries was measured and the results can be seen in Table 2. In this table, a comparison is made between a native ontology-based system, as was the case in the Dutch Network Statement Checker repository and a legacy system integrated using a mapping engine in between, namely D2R-server, as was the case for the Belgian data. The same dataset was used in both measurements, however, in order to be able to correctly compare both approaches. Each query was issued 10,000 times, and this resulted in an average execution time of 645.16 ms for query 1 in the case of the D2R approach, 414.64 ms in case of the native Jena approach and 6.19 ms for the legacy relational database approach. For query 2 these average execution times were respectively 1889.51 ms, 427.27 ms and 10.86 ms. For query 3 these were respectively 1796.62 ms, 429.50 ms and 6.67 ms. The minimum, maximum, average execution times as well as the standard deviation can be found in the Table 2.

All measurements were performed on an isolated Linux machine, running the Debian Etch distribution with kernel 2.6.17.14. It has an AMD Athlon(tm) 64 Processor 3000+ processor and 512 MB of RAM available. Java version 1.6.0 update 6 was installed. In the case of the D2R measurements, D2R-server version 0.4 was used. A comparison graph with these measurements can be seen in Figs. 13a and 13b.

The ontology version of the dataset contains 13 classes, 5 object properties, 68 data properties in the T-Box model, having an ALCIF(D) description logics expressivity. The A-Box contains 5055 individuals, resulting in 5749 class assertion axioms, 1093 object property assertion axioms and 27,170 data property assertion axioms. The relational database version of the dataset contains 13 tables. The two most important and data-intensive tables describing the network and the stations each contain respectively 133 rows by 5 columns and 561 rows by 6 columns.

We have also converted the queries into a non-ontology pure SQL query, in order to properly evaluate the overhead introduced by the ontology approach. It can be expected that by using the ontology-based integration mechanism, a considerable overhead is introduced, because of the addition of annotations and relationships. With this evaluation we want to present the overhead introduced by the ontology-based methodology. The longer times needed to process the same information in an ontology manner can be explained on the one hand by the required online conversion of the SQL results of potential suboptimal SQL-queries generated by the D2R mapping engine and on the other hand because of the description logics processing introduced by the pure Jena based approach. However, in the context of non-time critical applications, such as the Network Statement Checker, this extra overhead introduced by this approach is worth the effort compared with the increased transparency and adaptability of the suggested approach towards future systems.

Additionally, we believe that, given recent commercial initiatives, the performance of the ontology processing tools will improve. Moreover, given the reduced development effort and potential reuse supported by the notion of domain modelling, the usage of ontology-based implementations in non-time critical applications is still a very good candidate. Its other characteristics, such as the foundation in formal descriptions logics and the extensibility, support its usage in such distributed information integration applications even more.

Despite the fact that the ontology-based approach does introduce a certain overhead in the processing of the queries, this should not be an issue for non real-time critical applications. Moreover, the added value of using a domain model instead of an application specific model will make sure that the models will not change frequently. After all, once a common agreed view on a particular domain is reached, this is not likely to change very often as is the case with application specific models.

6. Transition from research demonstrator towards domain-wide commercial deployment

The biggest challenge for domain-wide adoption is arguably the creation of the ontologies, describing the domain of which the railway undertakings' systems provide data. These models need to be well-defined and formally proofed as in a semantic environment and in contrast to database schema, they do not only define the syntax of the data, but also the meaning of that data. A flawed model will lead to flawed integration and thus false decision making.

The addition of new ontological models and the alteration of existing ontologies should be coordinated by a standardization body as well. This is of paramount importance to ensure that these additions and alterations do not turn the existing models inconsistent. After all, the injection of ill-formed new information and knowledge could potentially corrupt the already existing deployments. However, using the hybrid ontology approach, as presented in Section 4.1, some of these issues can be prevented more easily. After all, the core of the ontology should be strictly coordinated and enforced in a domain-wide adoption and should be the responsibility of the standardization body while the additional domain ontologies could be used in a non-standardized and proprietary deployment. All these models can co-exist together, without the proprietary domain ontologies influencing the core ontology used by all stakeholders.

InteGRail proposed the notion of a core ontology, which describes the general railway concepts and its relations. Specific domain models can be attached to this core ontology, by means of subclasses and extensions of existing concepts. Additionally, new concepts and relationships can be created as an extension of this core model. In this way, more specific domain knowledge is inserted in the general model, and the existing applications, when reasoning is enabled, can take this extended information into account. This is similar to the conceptual description of the examples given in Section 3.2. Fig. 14 presents in an abstract way the notion of the core ontology, with its possible extensions with more specific domain ontologies. The core ontology is constructed out of the general concepts, common and generally accepted in the entire

Table 1
Queries used in the evaluation phase of the Network Statement Checker information integration platform.

Query 1	PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

LineNetworkEdgeQuery	PREFIX geo:<http://pervasive.semanticweb.org/ont/2004/06/geomeasurement>
	PREFIX nso:<http://www.owlontologies.com/NetworkStatementCheckerOntology.owl#>
	PREFIX core:<http://www.integrail.info/ont/SP3A.owl#>
	SELECT DISTINCT ?name ?namenode1 ?namenode2 ?lat1 ?lng1 ?lat2 ?lng2
	WHERE {
	?line rdf:type core:LineNetworkEdge
	?line core:startsAt ?nod1
	?line core:endsAt ?nod2
	?line nso:hasName ?name
	?nod1 nso:hasName ?namenode1
	?nod2 nso:hasName ?namenode2
	?nod1 nso:hasCoordinate ?c1
	?nod2 nso:hasCoordinate ?c2
	?c1 geo:latitude ?lat1
	?c1 geo:longitude ?lng1
	?c2 geo:latitude ?lat2
	?c2 geo:longitude ?lng2
	}

Query 2
InterconnectingNodeQuery
	PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX geo:<http://pervasive.semanticweb.org/ont/2004/06/geomeasurement>
	PREFIX nso:<http://www.owlontologies.com/NetworkStatementCheckerOntology.owl#>
	PREFIX core:<http://www.integrail.info/ont/SP3A.owl#>
	SELECT DISTINCT ?namenode ?lat ?lng
	WHERE {
	?line rdf:type core:LineNetworkEdge
	?line core:startsAt ?nod1
	?line core:endsAt ?nod2
	?line nso:hasName ?name
	{?nod1 nso:hasName ?namenode.} UNION {?nod2 nso:hasName
	?namenode.}
	?nod1 nso:hasCoordinate ?c1
	?nod2 nso:hasCoordinate ?c2
	?c1 geo:latitude ?lat1
	?c1 geo:longitude ?lng1
	?c2 geo:latitude ?lat2
	?c2 geo:longitude ?lng2
	}
	ORDER BY ?namenode

Query 3
TrackCharacteristicsQuery
	PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
	PREFIX geo:<http://www.pervasive.semanticweb.org/ont/2004/06/geomeasurement>
	PREFIX nso:<http://www.owlontologies.com/NetworkStatementCheckerOntology.owl#>
	PREFIX core:<http://www.integrail.info/ont/SP3A.owl#>
	SELECT DISTINCT ?line ?name ?namenode1 ?namenode2 ?lat1 ?lng1 ?lat2
	?lng2 ?voltage ?system ?axleload
	WHERE {
	?line rdf:type core:LineNetworkEdge
	?line core:startsAt ?nod1
	?line core:endsAt ?nod2
	?line nso:hasName ?name
	OPTIONAL { ?line nso:hasVoltage ?voltage.}
	OPTIONAL { ?line nso:hasSafetySystem ?system.}
	OPTIONAL { ?line nso:hasAxleLoad ?axleload.}
	?nod1 nso:hasName ?namenode1
	?nod2 nso:hasName ?namenode2
	?nod1 nso:hasCoordinate ?c1
	?nod2 nso:hasCoordinate ?c2
	?c1 geo:latitude ?lat1
	?c1 geo:longitude ?lng1
	?c2 geo:latitude ?lat2
	?c2 geo:longitude ?lng2
	}

railway domain. Some of these common concepts are, *e.g.* rail, sleeper, vehicle, locomotive, station or timetable. However, the detailed characteristics of a certain type of vehicle will be modelled as an extension of the core ontology in so-called domain ontologies. This can be the exact number of doors in this vehicle or the subsystems and their relationships on-board this vehicle.

Table 2
Measurement results for the processing of the 3 queries on the Network Statement Checker information integration platform.

	Query1(D2R) (ms)	Query1(Jena) (ms)	Query1(DB) (ms)	Query2(D2R) (ms)	Query2(Jena) (ms)	Query2(DB) (ms)
AVG	645.16	414.64	6.19	1889.51	427.24	10.86
MIN	640	382	6	1867	423	10
MAX	1140	827	50	2287	582	13
STDDEV	5.86	4.6	0.59	6.33	2.41	0.45

	Query3(D2R) (ms)	Query3(Jena) (ms)	Query3(DB) (ms)
AVG	1796.62	429.5	6.67
MIN	455	425	6
MAX	1935	444	9
STDDEV	16.12	1.51	0.54

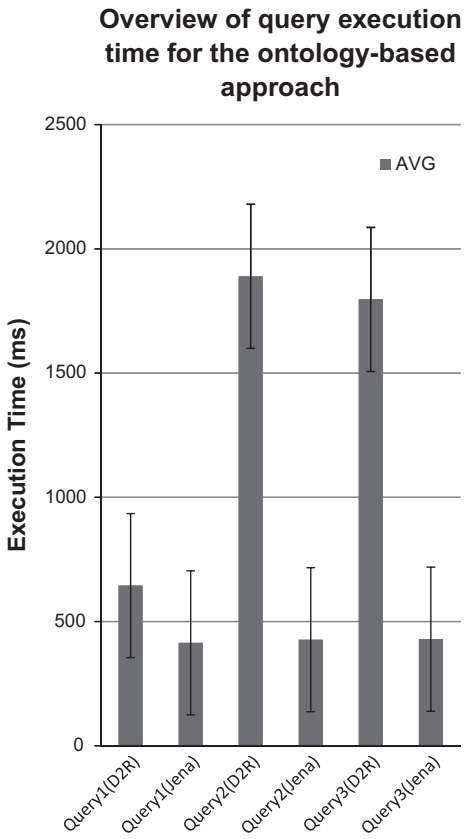

Fig. 13a. Overview of Query Execution Times, plotting the average time in ms for the ontology-based approach. The standard deviations are shown as well.

As an upcoming standard within the railway domain, the Technical Specification for Interoperability regarding Telematic Applications for Freight (TAF-TSI) and the Technical Specification for Interoperability Traffic Operation and Management Subsystem (OPE-TSI) requires that especially the Railway Undertaker (RU) should be informed if there are restrictions on the infrastructure.

"On the 18th of January 2006, the Technical Specification for Interoperability regarding Telematic Applications for Freight (TAF-TSI) regulation has entered into force.
This European regulation requires that the European railway industry develops and implements common standards to increase the interoperability of information, i.e. to facilitate the exchange of information between companies regarding rail freight services, notably as far as cross-border services are concerned. The regulation does NOT require replacing the existing IT systems of Infrastructure Managers (IMs) and RUs. It essentially requires that the interfaces between individual IT systems use a common language and follow certain specifications. The political intention behind the regulation is to boost the quality and productivity

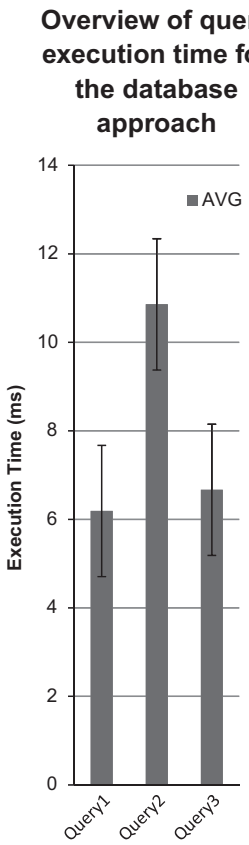

Fig. 13b. Overview of Query Execution Times, plotting the average time in ms for the relational database approach. The standard deviations are shown as well.

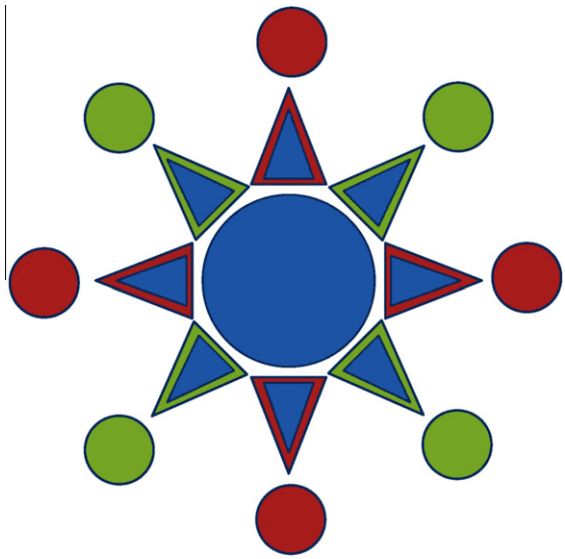

Fig. 14. The core ontology (in the center) extended with domain ontologies (on the outside).

of rail freight (notably for cross-border services) in Europe, in the context of an increasing road competition." (Strategic European Deployment Plan for the European-wide implementation of the technical specification for interoperability telematic applications for freight, 2006).

A number of such TSIs are currently being defined. We believe that the ontology-based approach as presented in this paper satisfies the requirements described in these specifications and could serve as a technical means to implement the requirements posed in the TSIs. The evaluation of the Network Statement Checker in Section 5 shows how the ontology mechanisms can be used to include existing information systems into an ontology-based information sharing platform.

Other business domains, with wide-scale deployment of semantic technologies, are law enforcement in the area of intelligent querying across various independent and unrelated systems and intelligent recognition and filtering (Gottschalk, 2007). A second example is supporting the Logistics sector in the area of planning, control and exchange between independent planning entities (Metatomix Inc, 2009a,b; Karageorgos et al., 2003). A third example can be found in Medicine, where the Semantic Technology supports the field of decision support systems (Metatomix Inc, 2009a,b).

7. Conclusions

In this paper we presented why data integration is an important problem, especially in the railway domain. A number of challenges were presented to successfully pursue such integration. On the one hand there is both the integration at syntactic and semantic level, but on the other hand there is also the issue that many stakeholders are rightly not willing to expose all data, but only certain parts of high-level not commercially sensitive information. Of all problems, heterogeneity is the most challenging one. In order to achieve semantic interoperability in a heterogeneous information system, the meaning of the information that is exchanged has to be understood across the systems. Thus, a common semantic model of the data in the different sources is required. Two distinct methods for expressing this common semantic model have been evaluated, namely UML and OWL. Their advantages and disadvantages were discussed as well.

We have studied in detail how this semantic model can be constructed by using one or more ontologies and what the advantages are of the different methods, how the ontologies can be used for more than information integration alone and how the mappings should be made between the ontologies and the real data. Mappings form an important aspect, as this mechanism facilitates the integration of existing legacy systems in an ontology-enabled integration environment. As such an additional interface, exposing a semantic view on the legacy data, is exposed. We also presented an overview of the different methods to construct an ontology model. We are convinced that the hybrid approach is the most suited approach for the railways. In this approach, a common core ontology should be created, standardized and maintained, modelling the main domain concepts and relationships, generally defined and accepted within the industry. Based on this core ontology, the systems to be integrated should be modelled. We evaluated this concept by an example application, the Network Statement Checker, which has been developed in the scope of the InteGRail project (InteGRail Consortium, 2009), a European Research project in the scope of FP6, and evaluated its performance, namely the Query Execution Times for three different typical queries. This exposed that the processing times for these queries is considerably slower than in a pure relational database approach. However, more and more commercial initiatives are emerging that provide ontology-based integration platforms and tools to manage the models and the integration on that platform. This will inevitably result in better performance in the coming years.

However, despite this current performance issue, we believe that OWL is the preferred model because in contrast to most information models, the baseline idea of ontologies is that their philosophy is to model the domain and not the application. Compared to other data integration mechanisms, the commonly agreed view of the data to be exchanged and integrated is the view of the domain and not the view of some commonly agreed application. More specifically OWL DL is based on formal description logics and allows for consistency to be checked and inference mechanisms to be defined inside the information model itself. The semantics, and not only the syntactic correctness, of the information are kept strictly together with the data itself. Additionally, because of the fundamental standards on which the models are constructed, generic tools can be used to process these ontologies. This is a very important aspect, as given the current interest in the Semantic Web, these tools are likely to become more and more powerful and thus better performing. We are convinced that the way forward for data integration is based on ontologies, because of the current research and development in this domain, their foundation in strict mathematics, adaptability and their suitability to be used in a distributed environment. Admittedly, the performance of these technologies in terms of processing times to answer queries is currently not as good compared to other techniques, such as relational database engines. However, an ontology model is more than just a data model. It is a common agreed view of a domain, which can easily be extended or adapted with domain rules for a specific application, in order to use reasoners to infer inherent information from the asserted data in the model. This facilitates the reuse of generic application models, because the logic can be transferred from the application towards the semantic model. Additionally, the shift towards using formal domain models instead of application specific data models, results in lower development costs, as the same model can be re-used and easily adapted in multiple applications. A number of modelling techniques have been presented in this paper. We believe that, given the complex structure of the railway domain, the hybrid approach is the best suited approach. An agreed vocabulary and common model is shared in this methodology, and this model is then extended with a proprietary model for the specific domain information systems to be integrated. This approach has been successfully validated by means of a demonstrator implementation, namely the Network Statement Checker. The ontology-based information integration approach formed the basis in this implementation. Given these characteristics, combined with their foundation in formal description logics, we are convinced that ontologies are an ideal mechanism to construct a heterogeneous data integration platform.

Acknowledgement

We would like to acknowledge that part of this research was supported by the European FP6 Research Project, InteGRail. Stijn Verstichel and Femke Ongenae would like to thank the IWT, Institute for the promotion of Innovation through Science and Technology in Flanders for their PhD grants.

References

Baclawski, K. et al, 2001. Extending UML to support ontology engineering for the Semantic Web. Lecture Notes in Computer Science, 2185.
Bechhofer, S., et al., 2004. Owl Web Ontology Language Reference, W3C Recommendation. <http://www.w3.org/TR/owl-ref/>.
Beckett, D., Berners-Lee, T., 2008. Turtle – Terse RDF Triple Language, W3C Team Submission. <http://www.w3.org/TeamSubmission/turtle/>.
Beckett, D., McBride, B., 2004. RDF/XML Syntax Specification (Revised), W3C Recommendation. <http://www.w3.org/TR/rdf-syntax-grammar/>.
Berners-Lee, T., 1998. Notation 3 (N3), A Readable Language for Data on the Web, N3 Specification. <http://www.w3.org/DesignIssues/Notation3.html>.
Bizer, C., Cyganiak, R., 2006. D2R-Server, publishing relational databases on the Semantic Web. In: Proceedings of the 5th International Semantic Web Conference. Athens, GA, USA, 5–9 November 2006.
Bray, T., et al., 2006. Extensible Markup Language (XML) 1.0 (fourth ed.), W3C Recommendation. <http://www.w3.org/TR/2006/REC-xml-20060816/>.
Bray, T., et al., 2006. Extensible Markup Language (XML) 1.1 (second ed.), W3C Recommendation. <http://www.w3.org/TR/2006/REC-xml11-20060816/>.
Buccella, A., Cechich, A., Brisaboa, N.R., 2003. An ontology approach to data integration. Journal of Computer Science and Technology 3 (2), 62–68.
Carroll, J.J. et al, 2004. Jena: implementing the Semantic Web recommendations. In: Proceedings of the 13th International World Wide Web conference on Alternate Track Papers & Posters, New York, NY, USA, 17–20 May 2004. ACM Press, pp. 74–83.
Computer Human Interaction & Software Engineering Lab (CHISEL) and its members, 2008. Jambalaya. <http://www.thechiselgroup.org/jambalaya>.
Clark & Parsia, LLC. 2009. Pellet, the Open Source OWL-DL Reasoner in Java. <http://pellet.owldl.com/>.
Cui, Z., O'Brien, P., 2000. Domain ontology management environment. In: Proceedings of the 33rd Hawaii International Conference on System Sciences. Island of Maui, Hawaii, 4–7 January 2000.
Ellson, J. et al, 2002. GraphViz: open source graph drawing tools. Lecture Notes in Computer Science 2265, 594–597.
EuRoMain Consortium, 2003. EuRoMain, European Railway Open Maintenance System. <http://www.euromain.org/>.
Falkovych, K., Sabou, M., Stuckenschmidt, H., 2003. UML for the Semantic Web: Transformation-Based Approaches. Knowledge Transformation for the Semantic Web.
Fernandez, M., Gomez-Perez, A., Juristo, N., 1997. METHONTOLOGY: from ontological art towards ontological engineering. In: Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Providence, Rhode Island, 27–31 July 1997, pp. 33–40.
Fonseca, F., Martin, J., 2007. Learning the differences between ontologies and conceptual schemas through ontology-driven information systems. Journal of the Association for Information Systems (JAIS) – Special Issue on Ontologies in the Context of IS 8 (2), 129–142.
Goh, C.H., 1997. Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources, Ph.D., MIT, USA.
Gottschalk, P., 2007. Knowledge Management Systems in Law Enforcement: Technologies and Techniques. Idea Group Publication, Hershey, PA.
Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5 (2), 199–220.
Grüninger, M., Fox, M.S., 1995. Methodology for the design and evaluation of ontologies. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, Quebec, Canada, 20–25 August 1995.
Guha, R.V., Brickley, D., McBride, B., 2004. RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation. <http://www.w3.org/TR/rdf-schema/>.
Haarslev, V., Möller, R., 2003. Racer: an OWL reasoning agent for the Semantic Web. In Proceedings of the 13th International Workshop on Applications, Products and Services of Web-based Support Systems, in Conjunction with 2003 IEEE/WIC International Conference on Web Intelligence, Hallifax, Canada, 13 October 2003, pp. 91–95.
Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A., 2008. RDF123: From Spreadsheets to RDF. International Semantic Web Conference, LNCS 5318, pp. 451–466.
Hart, L., et al., 2004. OWL-Full and UML 2.0 Compared. OMG TFC report.
Heflin, J., Hendler, J., Luke, S., 1999. SHOE: A Knowledge Representation Language for Internet Applications. Technical report CS-TR-4078 (UMIACS TR-99-71), Department of Computer Science, University of Maryland at College Park.
Horridge, M., 2005. OWLViz: Graph View of the Ontology. <http://www.co-ode.org/downloads/owlviz/>.
Horrocks, I., Patel-Schneider, P.F., van Harmelen, F., 2003. 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 (1), 7–26.
InteGRail Consortium, 2008. The Network Statement Checker. <http://www.integrail.eu/NetworkStatementWebApp.html>.
InteGRail Consortium, 2009. InteGRail, Intelligent Integration of Railway Systems. <http://www.integrail.info/>.
Kalyanpur, A. et al, 2006. Swoop: a web ontology editing browser. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 4 (2), 144–153.
Kalyanpur, A., Parsia, B., Sirin, E., Grau, C., Hendler, J., 2007. SWOOP: A Hypermedia-Based Featherweight OWL Ontology Editor. <http://www.mindswap.org/2004/SWOOP/>.
Karageorgos, A. et al, 2003. Agent-based optimisation of logistics and production planning. Intelligent Manufacturing Systems, 113–118.
Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A., 2004. The protégé OWL plugin: an open development environment for Semantic Web applications. In: Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan 7–11 November 2004, pp. 229–243.
Kotis, K., Vouros, A., 2006. Human-centered ontology engineering: the HCOME methodology. Knowledge and Information Systems 10 (1), 109–131.
Logic, U., 2008. OWL Ontology Artifacts have Durable Value. <http://www.logicu.com/175.html>.
McGuinness, D.L., van Harmelen, F., 2004. OWL Web Ontology Language Overview, W3C Recommendation. <http://www.w3.org/TR/owl-features/>.
Metatomix Inc., 2009. The Final Mile Challenge, Making Semantics Work in the Enterprise, Metatomix Whitepaper. <http://www.metatomix.com/>.
Metatomix Inc., 2009. Semantic Web Solutions at Work in the Enterprise, TopQuadrant Whitepaper. <http://www.metatomix.com/>.
Nardi, D. et al, 2003. The Description Logic Handbook: Theory, Implementation and Applications. The Press Syndicate of the University of Cambridge, Cambridge, UK.
The Object Management Group (OMG), 2006. Object Constraint Language, OMG Available Specification, Version 2.0 [Online] (Published May 2006). <http://www.omg.org/docs/formal/06-05-01.pdf>.
The Object Management Group (OMG), 2007. OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.1.2, OMG Available Specification [Online] (Published November 2007). <http://www.omg.org/docs/formal/07-11-04.pdf>.
The Object Management Group (OMG). 2007. OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.1.2, OMG Available Specification [Online] (Published November 2007). <http://www.omg.org/docs/formal/07-11-02.pdf>.
The Object Management Group (OMG), 2009. UML Resource Page. <http://www.uml.org/>.
Ontology Systems, 2009a. OSS/CAD Case Study: South Africa's Leading Converged Service Provider Simplifies Management of Operational Systems. <http://www.ontology.com>.
Ontology Systems, 2009b. OSS/CAD Data Sheet: Ontology Systems OSS/CAD. <http://www.ontology.com>.
Pinto, H.S., Martins, J.P., 2004. Ontologies: how can they be built? Journal of Knowledge and Information Systems 6 (4), 441–464.
Prud'hommeaux, E., Seaborne, A., 2008. SPARQL Query Language for RDF, W3C Recommendation. <http://www.w3.org/TR/rdf-sparql-query/>.

Quatrani, T., 2000. Visual Modeling with Rational Rose 2000 and UML. Addison-Wesley Professional publishing, Reading (Mass.).
Racer Systems GmbH & Co. KG, 2009. Racer Pro. <http://www.racer-systems.com/>.
Reilly, J.P., Wilmes, J., 2008. Application Integration using the SID. TM Forum.
Serafini, L., Tamilin, A., 2005. Drago: distributed reasoning architecture for the Semantic Web. In: Proceedings of the Second European Semantic Web Conference (ESCW), Heraklion, Greece, 29 May–1 June 2005, pp. 361–376.
Sirin, E. et al, 2007. Pellet: a practical OWL-DL reasoner. Journal of Web semantics: Science, Services and Agents on the World Wide Web 5 (2), 51–53.
Stanford Center for Biomedical Informatics Research, 2009. The Protégé Ontology Editor. <http://protege.stanford.edu/>.
Storey, M.-A., Noy, N.F., Musen, M., Best, C., Fergerson, R., Ernst, N., 2002. Jambalaya: an interactive environment for exploring ontologies. In: Proceedings of the 7th International Conference on Inteligent User Interfaces, New York, NY, USA, 12–15 January 2003. ACM Press, p. 239.
Strategic European Deployment Plan for the European-wide implementation of the technical specification for interoperability telematic applications for freight (TAF-TSI) – Project No: 2005-EU-93008-S. 2006. Deliverable 2 – Definition of the Functional and Performance Requirements and of the Associated data Necessary to Deliver the TAF System, Version 1.0, 17/07/2006. <https://www.eurnex.com/forms/spip.php?article482>.
Tamilin, A., et al., 2006. DRAGO: Distributed Reasoning Architecture for Galaxy of Ontologies. <http://sra.itc.it/projects/drago/system-description.html>.
Uschold, M., King, M., 1995. Towards a methodology for building ontologies. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, Quebec, Canada, 20–25 August 1995.
Vrandevic, D. et al, 2005. The DILIGENT knowledge processes. Journal of Knowledge Management 9 (5), 85–96.
Wache, H., et al., 2001. Ontology-based integration of information – a survey of existing approaches. In: Stuckenschmidt, H. (Ed.), IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, Washington, USA, 6 August 2001, pp. 108–117.