Modello di Sistema di Comando e Controllo generalizzato

"Modello di Sistema di Comando e Controllo generalizzato"

Imad Zaza

Attività svolta nel progetto RAISSS: RAILWAY SIGNALLING: SAFETY AND SECURITY

POR CReO FESR 2007 – 2013, LINEA D'INTERVENTO 1.5.a - 1.6, BANDO UNICO R&S ANNO 2012, cofounded by Tuscany region

Referente: Paolo Nesi

Distributed System and Internet Technologies Lab
Distributed Data Intelligence and Technologies Lab
Department of Information Engineering (DINFO)
University of Florence

http://www.disit.dinfo.unifi.it

Version 2.0

EXECUTIVE SUMMARY

All interlocking systems have the same basic targets. Beyond a safe railway, they are all working to maximize the capacity at which they can operate their networks, minimize passenger and freight delays, maximize the reliability of the infrastructure and rolling stock, and do all of these at minimum cost.

In the last two decades, railways have improved their safety grade through a series of engineering and process improvements.

But over time the level of improvement that can be achieved reaches a threshold beyond which further improvement by these means is minimal due loosely coupled of knowledge between different railways operators.

Also, the separation of railways into Infrastructure Managers and operating companies or Railway Undertakings (RUs) (in Italy we have for example RFI and Trenitalia respectively) means that there is a limit to the improvement that can be achieved if IMs and RUs cannot work together effectively.

The present paper summarize the work conducted by DISIT Laboratory to define a new interlocking experimental model based on an ontological model of the railway infrastructure, showing how it is able to support all main feature of the railway signaling systems, integrating existing systems and paving the way for a new generation of systems.

During the project lifetime a number of example applications were developed, to demonstrate that the concepts developed in the project work.

These example applications were demonstrated at the end of the project, to prove that the platform, the architecture and the common language strategy work and to prove that a real improvement of railway performance can be achieved by using this way to manage and share information.

Keyword lists

Railway signalization; Interlocking system design; SIL4 systems; Ontology; rdf; rdf schema; railML

Summary

Introduction

In the deliverables “Report of a comparative analysis of the Interlocking Systems” and “Methods, notation and tools for modeling Command and Control Systems” emerged that nowadays the approach of developing interlocking software are still based on translating control tables or circuit plan.

A new core interlocking logic, beyond to be EN58128 certified, must enhances the maintenance issue; in the case of railway domain this is translated in minimize the deployment for a specific track layout.

This minimization is obtained exploiting the interoperability of the system relatively to country rules and loosely-coupling with the track side devices as turnouts, level barriers, train detector etc.

As far as we saw in the above deliverables, the interlocking core software model are developed using an OOP approach which is for a reluctant domain i.e. railways just a revolution.

However, OOP didn’t resolve the interoperability issue natively and this is proved by the study in the relative derivable we shown.

A prominent way could be the setup of an expert system.

State of the art

To motivate the technology adopted in our interlocking model, we must show here the state of the art in resolving the data integration problem which is the main factor to maximize interoperability.

Euro-Interlocking

According to [1] which is also the main founder, the Euro-lnterlocking Data Preparation work was established to promote the standardization of data exchange file formats for interlocking applications.

The Euro-Interlocking Project has contributed to the development of standardized file formats for interlocking data exchange. These standardized means of exchanging information between different phases of the data preparation process are needed to fulfil the CENELEC [2] process for safety and reliability. Thus the file format standards describe a common European structure and framework for project and configuration data transfer between data preparation tools and for data exchange between railways and suppliers.

We found that this formalism it is not flexible enough to be updated, also, the Euro-Interlocking project hardly allows integration of further new requirements.

InteGRail

InteGRail [3] l is the first large project relating the railway domain that is based on using an ontology. It was launched in the 2005 and terminated in 2009.

The aims of this project is a new efficient information sharing in the railway world [4] which improve decision making which also will improve performance.

Moreover the consortium claims that will be possible to identify what information has to be shared and assure that the right information can obtained (we recall that the main problem, among others, is that the railway operators are sometimes unwilling to share knowledge to protect their know-how).

An instance of the project was a network statement checker introduced in [5] where they also provided a web app 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.

An interesting design concepts which we used in modeling is the conceptualization of railway line and routes.

INESS

In the INtegrated European Signaling System (INESS) project [3], which launched in October 2008 and ended in March 2012, various data ex-change tools was studied and prove that RailML has most potential.

The key value of the project is a comprehensive set of interlocking specification requirements.

Academic works

An interesting academic work on ontology applied on railway domain is presented in [6] and [7]. The ontology presented are focused on solving the problem of verification of a railway infrastructure by the point of view of planning optimization of a new railway infrastructure. In [7] they propose a 2-tier ontology hard based on the RailML infrastructure sub schema and completed with German national rule expressed in SWRL.

Despite the conclusion where they affirmed that generally an ontology approach on railway domain for verification is inconvenient, they exploited more verification cases in [7].

Ontologica

In work [8] was developed "Ontologica" system, a project born from a joint effort between the Department of Informatics, Bioengineering, Robotics and System Engineering of Genoa university and Ansaldo STS (from the same city) in the design of advanced information systems.

The aim of the project was twofold:

- The adoption of ontologies to manage the Centralized Traffic Control (CTC) logics of a railway system;
- The improvement of the user interface through the exploitation of natural language queries.

The ontology is used to translate “Operator experience know-how” in a formal way without the capacity to infer new knowledge.

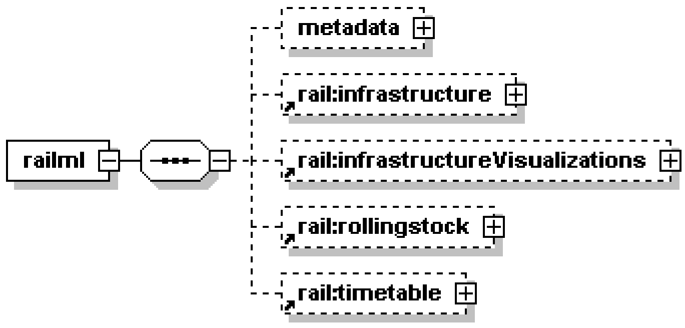
RailML

To represent a railway track layout one of the most prominent is a specific xml schema: RailML. As stated in [9] The RailML.org Initiative was founded in early 2001 as a development partnership of railways companies and research institutes located in different countries which have developed some interface tools for simulations on timetables, and have promoted an approach permitting a re-use of these tools for different contexts and operators. The RailML is an XML schema composed of three subschema: Infrastructure, Rolling Stock and Timetable.

The infrastructure schema describes track positions according to nodes and branches modelling.

The rolling stock sub-schema represents the characteristics of vehicles moving on the aforementioned tracks which are di-vide also in ones that propel themselves and ones that cannot. Typically in such schema it will find static data for both vehicles as vehicle length, mass, gauge, type, country acceptance certificates and so on. The vehicles that can propel themselves distinguish themselves by acceleration, deceleration and safe-ty system characteristics.

The timetable schema shows a realizable pattern of train movement over a section of track. Realizable means that the trains will not physically interact. The timetable therefore reflects the sequence of trains that use the infrastructure as described in their associated schemas.



All sub-schemas are based on the approach of defining interfaces where object descriptions are based on the XML syntax, i.e. to specify data format by XML schema. We appoint that, being a standard each sub-schema may not contain country specific attributes which make RailML files compatible with each other. Also, being svg an XML-based vector image format for two-dimensional graphics it is possible to implement via xslt scripts [18] a graphic representation of a track layout.

However, being an XML, RailML does not contains semantic information that is an XML instance may be a valid file but it could be a representation of a nonsense layout of a station.

We focused on an infrastructure subschema of railML in which the representation of physical elements concerning the railway route resides.

The element definitions range from tracks, signals and level crossings to security related parts like train detection circuits or transponders for train deceleration and many more.

RailML infrastructure subschema

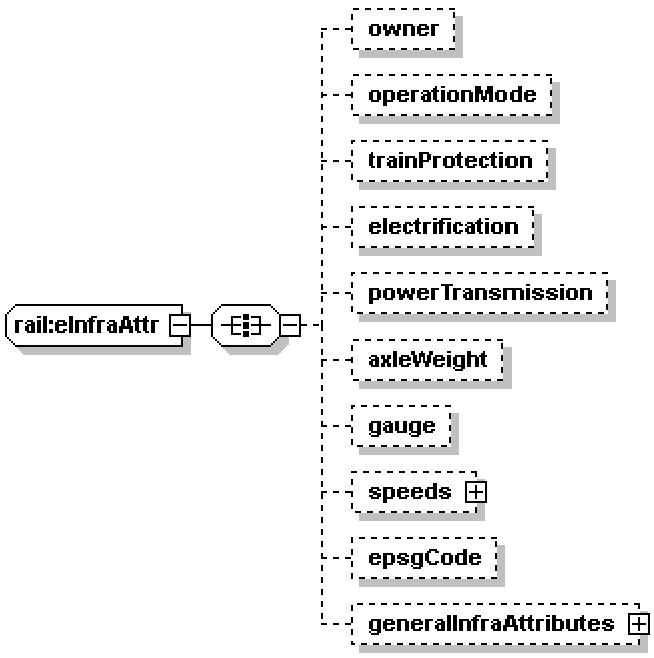
An interlocking area of interest is described in railml as the following code.

```
<railml>
        <infrastructure>
                </infraAttributes>
                <tracks>
                        <trackTopology>
                                <trackBegin>
                                        </connection>
                                </trackBegin>
                                <trackEnd>
                                        </connection>
                                </trackEnd>
                                <connections>
                                        </switch>
                                </connections>
                        </trackTopology>
                        <trackElements>
                                </gradientchanges>
                                </speedchanges>
                                </levelCrossing>
                                </bridges>
                                </tunnels>
                        </trackElements>
                        <ocsElements>
                                </signals>
                                </trainDetectionElements>
                                </balises>
                                </ocsElements>
                </tracks>
                <trackGroups>
                        <line>
                                </trackRef>
                        </line>
                </trackGroups>
        </infrastructure>
</railml>
```

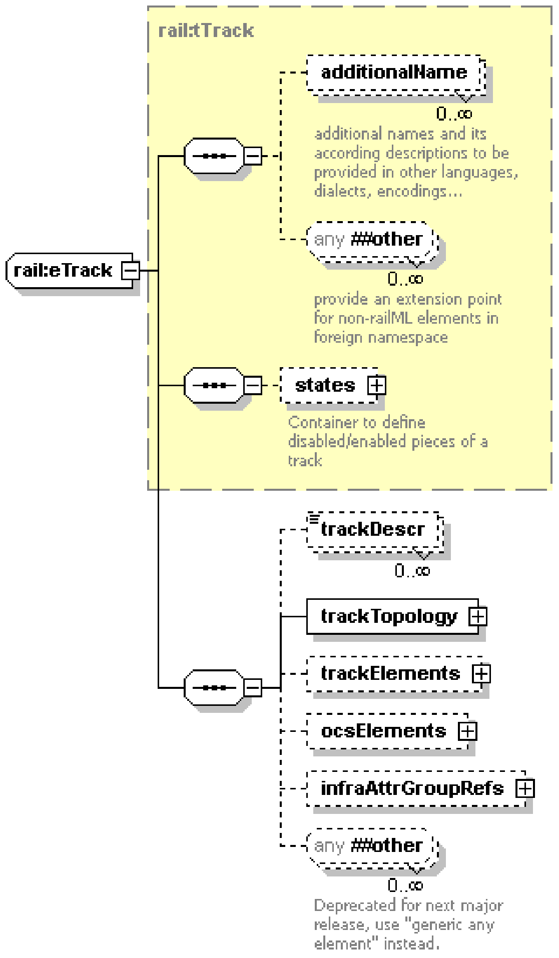
The root element in the RailML infrastructure sub-schema contains six sub elements:

- InfraAttrGroups
- Tracks
- Trackgroups
- OperationControPoints
- Controllers
- SpeedProfiles

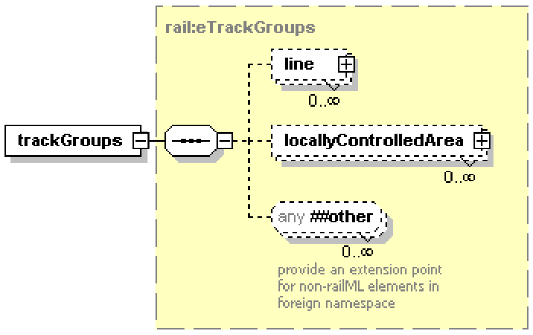
InfraAttrGroups element describes some general characteristics of the railway network in question and may be filed for the purposes of recognition.



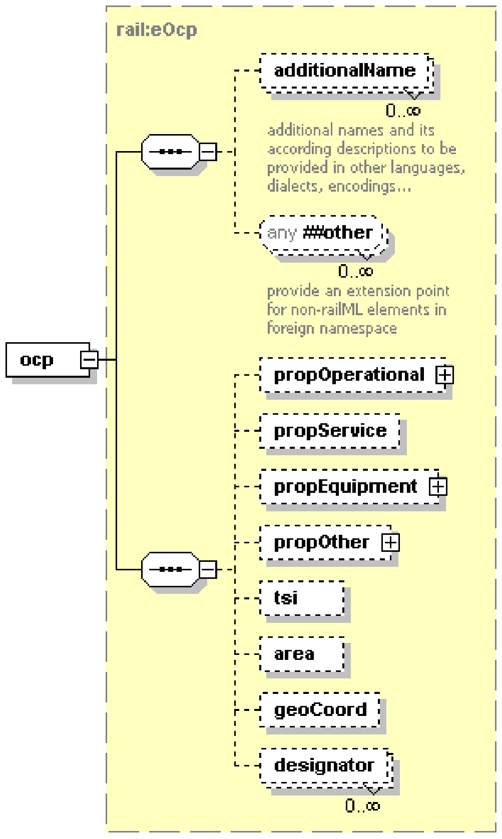
The Tracks sub element describes the position of track and the related elements on that track like signals. Furthermore, the tracks sub element also describes how multiple sections of track connect.


rail:tTrack
additionalName
0..∞
additional names and its according descriptions to be provided in other languages, dialects, encodings...
any ##other
0..∞
provide an extension point for non-railML elements in foreign namespace
states
Container to define disabled/enabled pieces of a track
rail:eTrack
trackDescr
0..∞
trackTopology
trackElements
ocsElements
infraAttrGroupRefs
any ##other
0..∞
Deprecated for next major release, use "generic any element" instead.

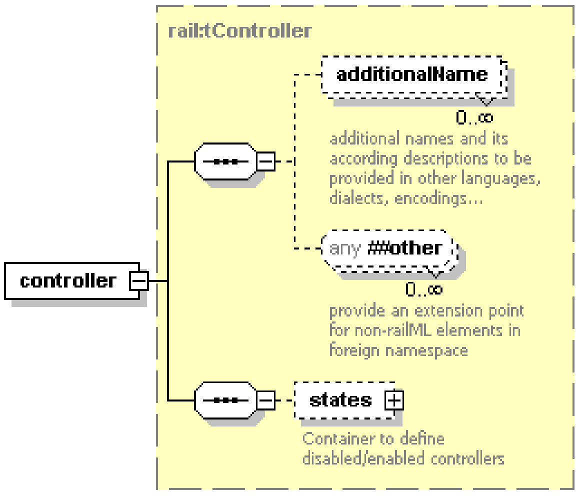
Trackgroups element aims to categorize tracks per corridor. This infrastructure sub element might be useful to identify the relations between station layouts.



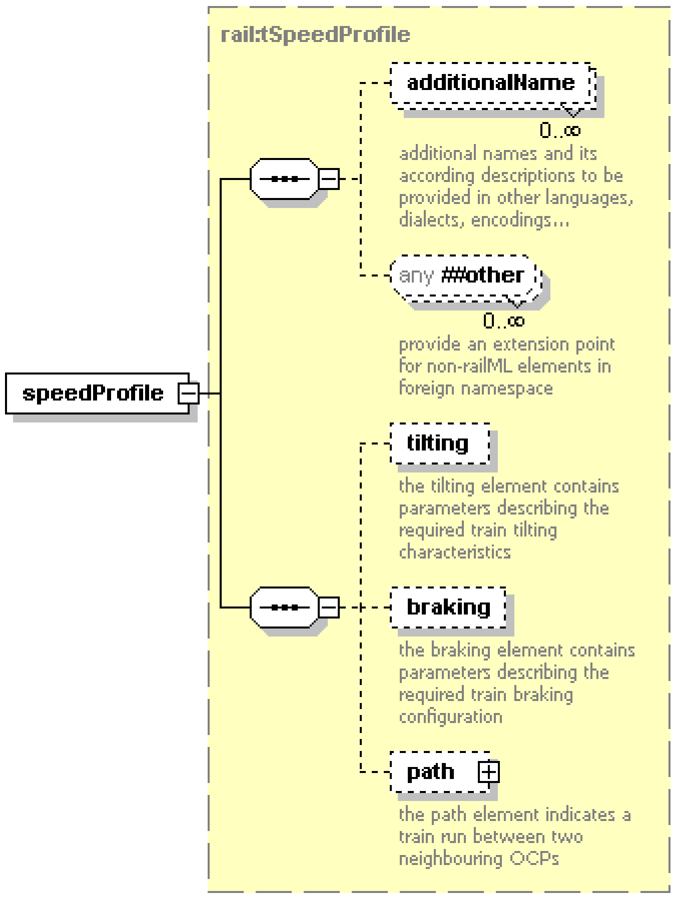
OperationControlPoints group the infrastructure elements for operational reasons of traffic control.


rail:eOcp
ocp
additionalName
0..∞
additional names and its according descriptions to be provided in other languages, dialects, encodings...
any ##other
0..∞
provide an extension point for non-railML elements in foreign namespace
propOperational
propService
propEquipment
propOther
tsi
area
geoCoord
designator
0..∞

Controller define rail operation facilities positioned along the track.


rail:tController
controller
additionalName
0..∞
additional names and its according descriptions to be provided in other languages, dialects, encodings...
any ##other
0..∞
provide an extension point for non-railML elements in foreign namespace
states
Container to define disabled/enabled controllers

The Speed Profiles sub element encompasses the parameters that determine the train's speed.


rail:tSpeedProfile
speedProfile
additionalName
0..∞
additional names and its according descriptions to be provided in other languages, dialects, encodings...
any ##other
0..∞
provide an extension point for non-railML elements in foreign namespace
tilting
the tilting element contains parameters describing the required train tilting characteristics
braking
the braking element contains parameters describing the required train braking configuration
path
the path element indicates a train run between two neighbouring OCPs

Knowledge–base systems

From the aforementioned discussion we established that in the railway domain:

- A key of improvement is “standardization towards interoperability”;
- RailML is an ongoing format of data exchange;
- Knowledge base engineering is experimented;

In the domain of industrial engineer, better knowledge can be more important for solving a task than better algorithms.

By the definition of Feigenbaum [10] Knowledge-based system or commonly expert systems (ES) are computer programs based on Artificial Intelligence (AI) techniques and designed to reach the level of performance of a human expert in a limited problem solving domain.

An ES [11] usually consists of four major components:

- a database;
- a knowledge base;
- an inference engine;
- a user interface.

The data base contains the data used by the ES. The data may be a part of the program, it may be part of a data base, or it may be elected form the user during the use of the system. This is normally the same data that a human expert uses to solve the problem.

The knowledge base contains the knowledge that the ES uses to process the data, typically the domain-specific knowledge an expert would use to solve the problem. Generally, this knowledge can be represented in a number of ways; it may be rule based or frame based. Rule based knowledge representation generally takes the form of:

```
If cond then consequence
```

Often a weight is used to represent the degree of confidence the probability or the strength of a rule. Frame based knowledge representation uses a frame to capture the characteristic that define the knowledge about the entity that is of interest in the application. Typically frame describe class of objects. The frame generally consists of a collection of “slots” that describe characteristic of the objects. These slots may then be filled with other frames describing other objects.

Until the last decade, the main approach to build an expert system relies on prolog or lisp languages dialect.

New way of expertizing systems: Rise of the Semantic Web

The Web was designed as an information space, with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help.

One of the major obstacles to this has been the fact that most information on the Web is designed for human consumption, and even if it was derived from a database with well-defined meanings (in at least some terms) for its columns, that the structure of the data is not evident to a robot browsing the web.

Leaving aside the artificial intelligence problem of training machines to behave like people, the Semantic Web approach instead develops languages for expressing information in a machine process able form.

The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. The first steps in weaving the Semantic Web into the structure of the existing Web are already under way. In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present.

A keystone of semantic web is the ontology term. Nowadays, ontologies enrich the knowledge engineering. According to [12] the term "ontology" can be defined as an explicit specification of formal shared conceptualization:

- A conceptualization, in this context, refers to an abstract model of how people think about things in the world, usually restricted to a particular subject area.
- An explicit specification means that the concepts and relationships in the abstract model are given explicit names and definitions. The name is a term, and the definition is a specification of the meaning of the concept or relation. A definition says how a term necessarily relates to other terms.
- Formal means that the meaning specification is encoded in a language whose formal properties are well understood—in practice, this usually means logic-based languages that have emerged from the knowledge representation community within the field of Artificial Intelligence. Formality is an important way to remove ambiguity that is prevalent in natural language and other informal notations; it also opens the door for automated inference to derive new information from the meaning specifications.
- Shared means that the main purpose of an ontology is generally to be used and reused across different applications and communities.

There has been much discussion on what exactly counts as an 'ontology'; however there is a common core that runs through virtually all approaches:

- a vocabulary of terms that refer to the things of interest in a given domain;
- Some specification of meaning for the terms, [ideally] grounded in some form of logic.

Ontologies represent many different kinds of things in a given subject area (e.g. wing, physical object, wire). These things are represented in the ontology as classes (sometimes called concepts) and are typically arranged in a lattice or taxonomy of classes and subclasses.

Each class is typically associated with various properties (also called slots or roles) describing its features and attributes as well as various restrictions on them (sometimes called facets or role restrictions).

An ontology together with a set of concrete instances (also called individuals) of the class constitutes a knowledge base. The lattice or taxonomy of classes is a primary the focus of most ontologies

What distinguishes different approaches to ontologies is the manner of specifying the meaning of terms. This gives rise to a kind of continuum of kinds of ontologies.

At one extreme, we have very lightweight ones that may consist of terms only, with little or no specification of the meaning of the term. Conversely, at the other end of the spectrum, we have rigorously formalized logical theories, which comprise the ontologies. As we move along the continuum, the amount of meaning specified and the degree of formality increases (thus reducing ambiguity); there is also increasing support for automated reasoning.

Many different representation formalisms have been explored, and reasoning engines developed. A key result is the proven existence of a tradeoff between representational power of a language (i.e. the ability to represent/express many different kinds of knowledge) and the efficiency of the reasoning engines.

In one sense, ontologies are a sub-area within knowledge representation. A major difference is that, unlike for Knowledge-Base in general, a key focus of ontologies is on knowledge sharing.

There is a growing interest in the idea of "Ontology-Driven Software Engineering" in which an ontology of a given domain is created and used as a basis for specification and development of some software.

The idea is to create an ontology that characterizes and specifies the things that the software system must address, and then use this ontology as a (partial) set of requirements for building the software.

The ontology is used as the basis for software development. For example, the development of an entire suite of product design software (including viewing and presentation tools, data bases, and even marketing and accounting tools that are used to track product sales) could be driven from the same ontology.

This would ensure easier interoperability among software systems whose relationships are typically only implicit.

When the ontology changes, the code is automatically updated. A large variety of applications may use the accessory functions from the ontology. Not only does this ensure greater interoperation, but it also offers significant cost reduction for software evolution and maintenance. A suite of software tools all based on a single core ontology are semantically integrated for free, eliminating the need to develop translators.

According to w3c, the semantic web framework is described as the following figure.

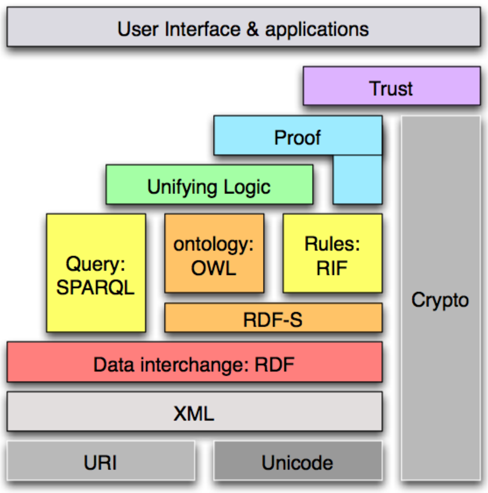


FIG. 1: W3C SEMANTIC WEB FRAMEWORK STACK

RDF, Resource Description Frame- work which is a system for expressing knowledge about things, or resources by identifying each resource with a URI or Unique Resource Identifier and by defining relationships between resources and/or explicit data values. Relationships are described by triplets of URIs where the middle URI functions as a predicate, defining the type of relationship between the other two URIs, which are referred to as subject and object respectively. By connecting together resources, an RDF graph is formed, which is a network of interconnected nodes, represented by the URIs.

The main format for storing RDF is and XML-format called RDF/XML. RDF/XML is not very easy to type or read though, and is not intended as a format for manipulating directly as a notation format. Instead, it is used as stable storage format by software, which can take advantage of the XML representation by using existing XML parsing libraries for managing the data.

While RDF provides a system for defining properties and relationships between entities, it provides no mechanism for describing these properties and relations. In order to describe the meaning of properties and relations, vocabularies have to be introduced that connect properties and relations with a defined meaning. It is not realistic though to collect all possible semantics in one single vocabulary. Instead, the W3C provides a language for expressing vocabularies that allows the creation of specialized vocabularies and ontologies within all kinds of fields as complements to more general standardized vocabularies. This language for describing vocabularies is called RDFS or Resource Description Framework Schema and contains predicates such as subClassOf, type, and isDefinedBy which can be used to define vocabularies.

We have mentioned ontologies, and RDFS can also be seen as a minimal ontology description language. It serves partly the same role as ontology languages such as OWL (described in the next section) while not being as comprehensive. The terms ontology and vocabulary have some overlap in their use.

For the construction of advanced ontologies, e.g. for precisely defining the meaning of railway terms and how they relate to each other, the semantics of RDFS is not enough. In fact, the expressivity of RDF and RDF Schema is deliberately very limited. Instead OWL, or Web Ontology Language, was created for defining ontologies of arbitrary complexity.

OWL has powerful features for expressing semantics by the use of logical operations such as union, disjointness etc.

OWL is available in three versions; OWL-Lite, OWL-DL and OWL-Full, with increasing degree of expressivity. This is because with increasing expressivity it is harder to create tools that provide full support for a language. OWL-Lite is created to enable most tool-makers to fully support at least a basic level of OWL, while OWL-DL is the language with enough expressivity for most ontology building tasks. DL stands for “Description Logics”.

The most expressivity is provided by OWL-Full but it is assumed that no tool will be able to completely support OWL-Full.

SPARQL is the W3C recommended language for querying of RDF data, or RDF graphs. It is much like SQL, the query language for relational databases, but for querying RDF graphs instead of database structures. In SPARQL, one expresses an RDF graph pattern which includes variables that should be bound, indicated with a question mark,

Reasoning

The semantic web way of storing knowledge, enables letting computers do more high-level knowledge processing tasks In addition to allowing integration and data querying, especially when using an expressive language such as OWL. One example is drawing conclusions that are not explicitly stated as facts but require several pieces of knowledge to be combined i.e. making implicit knowledge explicit.

The process of performing this kind of higher-level knowledge processing is often called reasoning and software that performs it are referred to as “reasoner”.

There are different approaches to reasoning and the expression of knowledge that can be used for reasoning. The main approach for reasoning within the W3C recommended standards is so called “Description Logics” supported by OWL-DL, which provides means to describe relationships between classes and instances, by using logical constructors. Examples of typical such constructors are disjointness, cardinality and transitivity. An OWL-DL reasoner typically allows operations such as validation, which is about checking that all expressed knowledge in a knowledge base are logically consistent with each other, classification, which is to assign a class membership to an entity, based on a class description expressed in OWL, and finally realization, which is to create instances, based on a class description.

RailOnto Ontology's domain

Here we will establish the domain of discourse emphasizing the relevant concepts.

Layout abstraction

A railways network is mainly composed of connecting tracks segment over which relies a logical one that describes the path between locations.

Each path correspond to what it is termed line. Such lines are topologically an abstract edges where nodes are called line node (i.e. stations).

A station identifies (controls) one or more interlocking areas. This areas are divided in block sections. The position of a rolling stock is realized by train detectors as track circuits, axle counter or the most recent balises.

A block section is a set of logically connected track sections that share the same train detector and is guarded by signals. A rail-road switch (or pair of points), is a special junction which enabling railway trains to be guided from one track to another.

These points can be moved laterally into one of two positions to direct a train coming from the narrow end toward the straight path or the diverging path. A train moving from the narrow end to-ward the point blades is said to be executing a facing-point movement.

Each one of the railway element described above, has certain attributes associated with it. For example, track sections are characterized by their gradient, maximum speed allowed and direction of motion (single or bidirectional).

A switch generally has a straight "through" track (such as the main-line) and a diverging route. Right-hand switches have a diverging path to the right of the straight track, when coming from the narrow end, and a left-handed switch has the diverging track leaving to the opposite side.

A route is an abstract set of track sections starting at one signal (entry signal) and ending at another signal (exit signal). An inverse route is such that route's starting and ending points are the ending and starting points of another route. When the routes cover different track segments but end at the same position they are called opposite routes. Any particular route required through a track layout consists of traversing a number of points in either a trailing or facing direction.

The movement of a train is regulated by signals which are established in front of blocks section, such fixed signals (generally used for the velocity) or two or three aspects semaphore lamps installed on track side which indicate the well-known actions (proceed, warning, stop) as in table below.

TABLE 1: SIGNAL ASPECT MEANINGS

Aspect	Meaning
Yellow	The next railway block is free, proceed with a predefined speed
Green	The next two railway blocks are free
Red	The next block is occupied, stop immediately

Yellow-Yellow	The next block is free, proceed with a track change
Yellow-Green	The next two blocks are free, proceed with a track change
Yellow-Red	A special color for entering to an occupied railway block
Flashing (Yellow, Green, Red)	Flashing signal lights are used while leaving an unsignalled railway block

Interlocking function

The interlocking system shall provide facilities that enable the signaler to:

- Request movement authorities
- withdraw movement authorities
- set signaling functions

For the purpose of our study the interesting part is related to requesting movement authorities.

A movement authority is an ensemble of track sections, turnouts and signals to be set and locked to permits a train conductor proceeding on requested route which is defined by entry and exit signal.

Route setting involves a collection of adjacent track circuits, points and signals. A route can be set and reserved for a passage of a train along this route. To assure the safety, firstly, the interlocking system verifies that the route does not conflict with other routes previously set. Secondly, the points along the route are locked in the correct positions. If the related points are not in the correct positions, the controller will attempt to set and lock them in the correct positions. Thirdly, the track circuits along the required route are all clear or unoccupied so that nothing obstructs the passage of the train. Then the entry signal can be cleared.
Thus requesting a route equals to query our knowledge-base to obtain the state of the involved element.

Knowledge-Base Interlocking (KBI)

Normally an interlocking model is then converted to Ladder Logic Diagram (LLD) , which can be implemented on a Programmable Logic Controller (PLC).

This is not the case of our model. Conversely our model could be used to exploit the control table of a track layout which then could be translated in Boolean equations to verify the accuracy and signalization design.

Architecture

A concrete instance of such model is showed above.

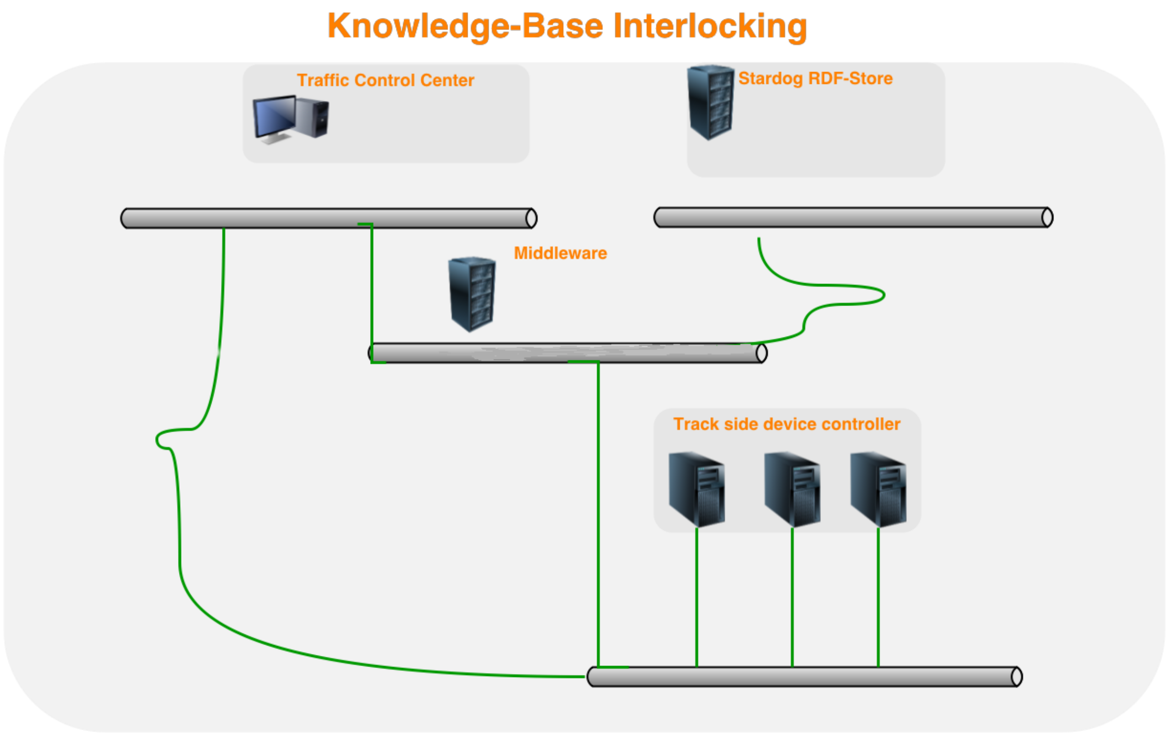


FIG. 2: GENERAL ARCHITECTURE

At traffic control center relies an application which monitor the states of the track devices.

Once a train request a route, this is translated in a set of SPARQL queries which are defined according to interlocking country rules and then sent to a middleware service.

The query result are then processed by the middleware service which inquire the rdf store and then communicating with the track devices and traffic control center application for the actions to be done.

The RDF store contains the railway station relative RDF which was previously translated from a RailML file.

It is also possible to insert in the RDF store the interlocking principles hard code in the control tables using XLST scripts [13] as showed in the figure.

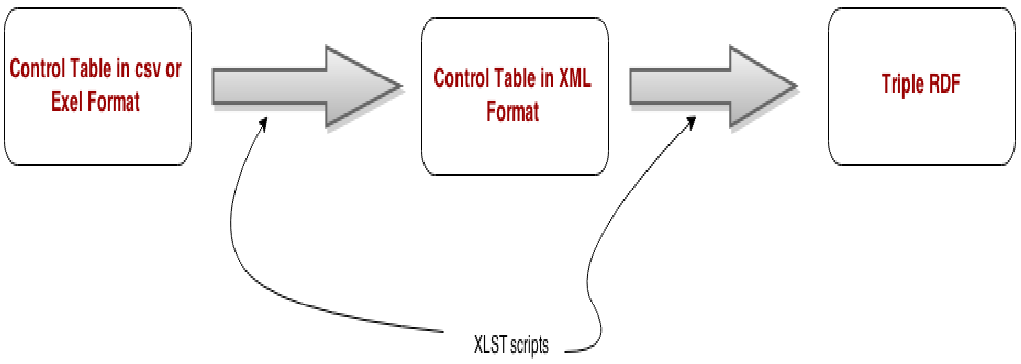

Control Table in csv or Exel Format
Control Table in XML Format
Triple RDF
XLST scripts

FIG. 3: TRANSFORMATION OF THE CONTROL TABLE TO RDF TRIPLE USING XSLT

The XML schema of the control table is shown in the following code:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="RouteLocks">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="route" type="xs:string"/>
        <xs:element name="tracks" maxOccurs="unbounded">
      <xs:element name="Points">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="normal" type="xs:string" maxOccurs="unbounded" />
            <xs:element name="reverse" type="xs:string" maxOccurs="unbounded" />
       </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="routeid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

</xs:schema>
```

As showed in the following figure the abstract system architecture is composed of three parts: a knowledge-base, input documents (RailML files, interlocking principles, specific rack elements field), and input and output real time signals belonging to track side devices. The knowledge base is composed by the ontology which will be presented soon and significant instances.

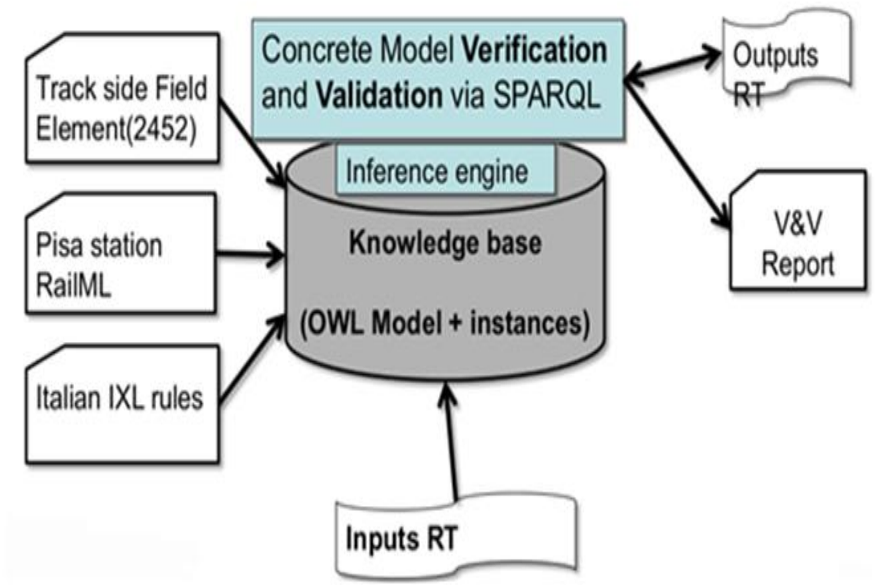


FIG. 4: ABSTRACT MODEL

Because we use a model driven developing philosophy it's possible to verify or validate general and functional requirements.

In the concrete model we give as input the real instance of the field elements, a specific railway and specific set of interlocking rules.

Ontology Developing

We used RailML infrastructure sub schema as a well-established domain of interest specification identifying elements and their characteristic which constitute our embryonic prototype ontology. This lead manually translate from the XML schema to ontology.

The data model of a RailML describes a node labeled tree, while OWL's data model is based upon the subject-predicate-object triples from RDF as we exposed in the above sections.
RDF-Schema defines a vocabulary for creating class hierarchies, attaching properties to classes and adding instance data.
As we discussed above, OWL on the top of RDF and RDFS, restrictions, such as cardinality constraints on properties, can be expressed.

This enables the straightforward representation of relational data in OWL:

- relations/tables correspond to classes;
- Columns to properties and rows to instances.

But the detection of relational structures within XML is difficult. For instance, there is the general question, how to handle nested tags. On the one hand they can be considered representing a"part-of" relationship or they express a "subtype-of" relationship.

Due to focusing on data oriented XML, we assume a relational structure and use this implicit knowledge about the design of the source documents to improve the transformation process.

For nested elements, we have chosen a middle course: For the case, when one element contains another element, which contains not only a literal, we assume a"part-of" relationship, so that we can assume a 1: N relationship.

This is mapped to an owl:ObjectProperty, which establishes a relationship between two classes. But we also can create "subtype-of" relations, i.e. we link together named xsd:complexTypes and therefrom derived elements. In the following figure we show how we exploited "Signal aspect" complex element.

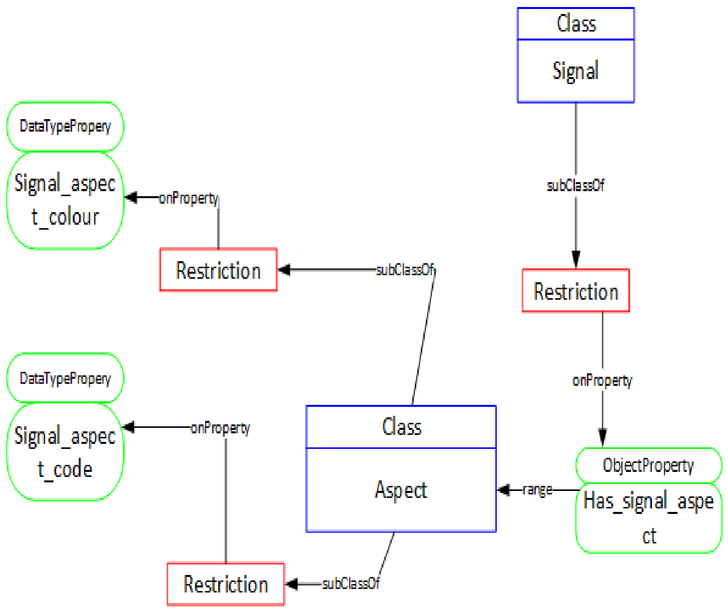


FIG. 5: SIGNAL ASPECT

Here multiple inheritance is possible (more than one domain). Classes (owl:Class) will emerge from xsd:complexTypes and xsd:elements according to the following rules: For the case, that an element in the source RailML tree is always a leaf, containing only a literal and no attributes, this element will be mapped to an owl:DatatypeProperty having as domain the class representing the surrounding element. RailML attributes will be handled equally, i.e. mapped to owl:DatatypeProperties, too.

Despite there is no real database counterpart for XML attributes and attributes are mostly used in document oriented XML, it makes sense to assume them representing database columns.

RailML Schema also contain arity constraints like xsd:minOccurs or xsd:maxOccurs, which we map to the equivalent cardinality constraints in OWL, owl:minCardinality and owl:maxCardinality.

All "name" attribute for each class is translated as subclass of dataproperty foaf:name. All attribute description for each class is translated as annotation in dc:description via protégé editor.

TABLE 2: CONVERSION RULES XML TO OWL

XML	OWL
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
named xsd:complexType	owl:Class
named xsd:SimpleType	owl:DatatypeProperties
xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sequence, xsd:all	owl:intersectionOf
xsd:choice	combination of owl:intersectionOf, owl:unionOf and owl:complementOf

This prototype derived directly form RailML was unsatisfactory for the following reasons:

- the ontology was ”flat” that is no inference at all could be obtained
- due the fact that XML has no semantic, explicit properties (i.e. connections) result in redundant information

The next stage of the engineering process encompass disassemble the concepts and their relation to assemble a new ontology reusing others we found suitable to describe the domain in question. This approach it is usual and recommended in ontology developing because it shares with object oriented methodology reusing of existing results. This is caused by the needing of the modelled system which could interact with other applications that have already committed to particular ontologies or controlled vocabularies.

Moreover, we identified two macro class each one related to the main aspects of interest: the network related concepts and the device related ones. Each one of the aforementioned is an extension of OTN and SSN respectively.

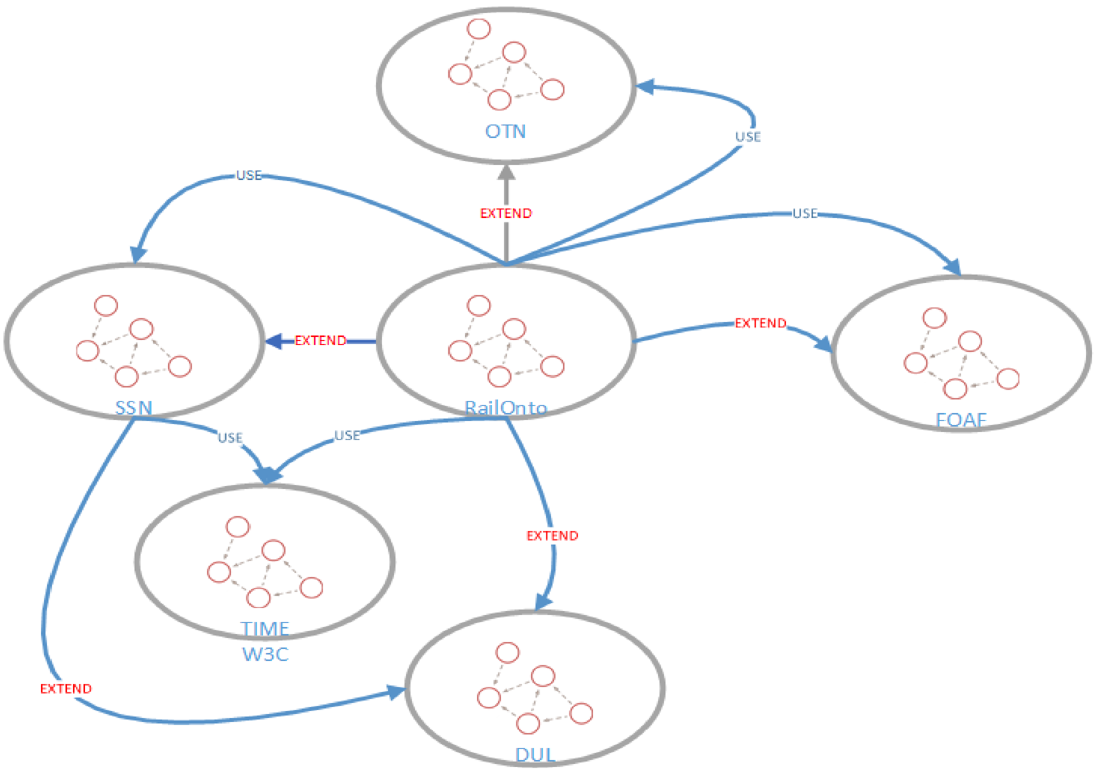


FIG. 6: ONTOLOGY MODULES

OTN (Open Transport Networks), which was defined within the framework of the REWERSE project [14], is a general purpose ontology oriented to transportation networks based on GDF (Graphic Data Format) which is a widely used binary data format for storing geographic data. OTN consists of five macro classes [14]:

- Feature contains all GDF features as OTN classes;
- Geometric defines the geometric forms of features;
- Composite Attributes represent classes consisting of composed attributes;
- Relationship describes the non-geometric relationships between features;

Transfer Point is a class which describes how to get from one object to another (e.g. train Stations).

Because OTN, describe an abstract graph via the *otn:node* and *otn:edge* concepts, we extend that introducing Railway edge and Railway node as middle class of Railway Element, Railway Line Edge, Railway Element Junction and Railway Line Node(see figure).

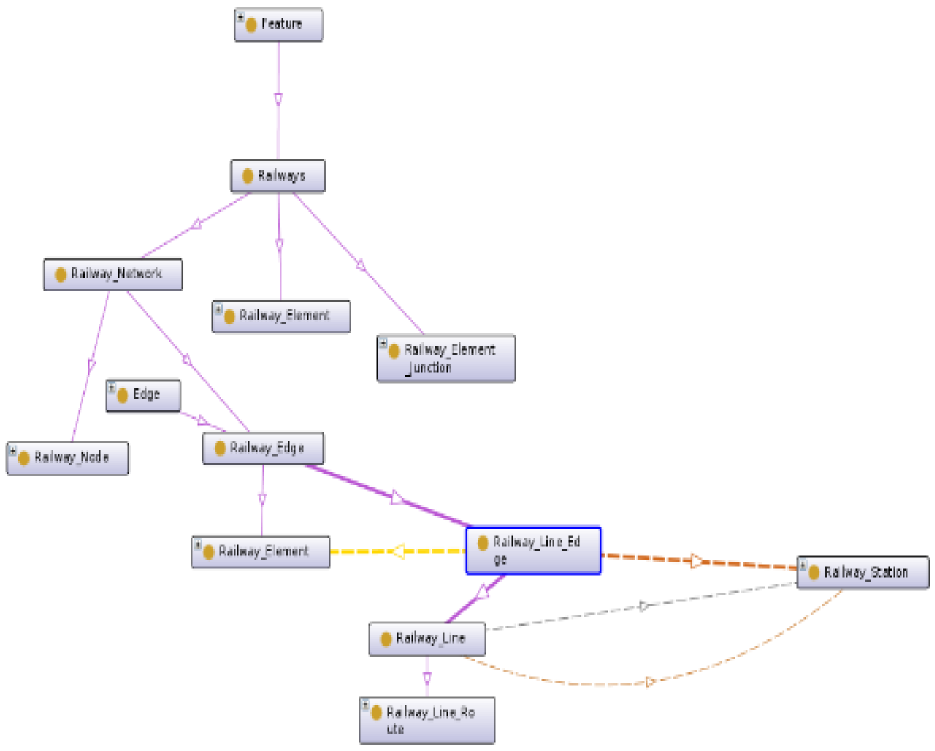


FIG. 7: RAILWAY LINE

Being *Track Section* a descendant of *otn:Edge*, is characterized by *otn:'starts at'* something and *otn:'ends at'* another thing. We needed to extend this OTN properties with symmetrical feature to be able to infer the presence or not of switches, hence introducing begins at and ends at object property.

As we pointed in the aims, we need also to query the semantic model to check if a requesting route by a rolling stock could be granted or not. Hence, it must model the train detector elements and signal respectively to check the track occupancy and to show the right aspect. To accomplish the above we reused SSN ontology. As stated in [15], The Semantic Sensor Network (SSN) ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment. It is noteworthy that the above mentioned ontology is built around an ontology design pattern describing the relationships between sensors, stimulus, and observations, the Stimulus- Sensor-Observation (SSO) pattern. For our purpose this is translated in extending *ssn:'Sensing Device'* and *ssn:Device* to describe the concepts of *Signal* and *Train Detection Element*.

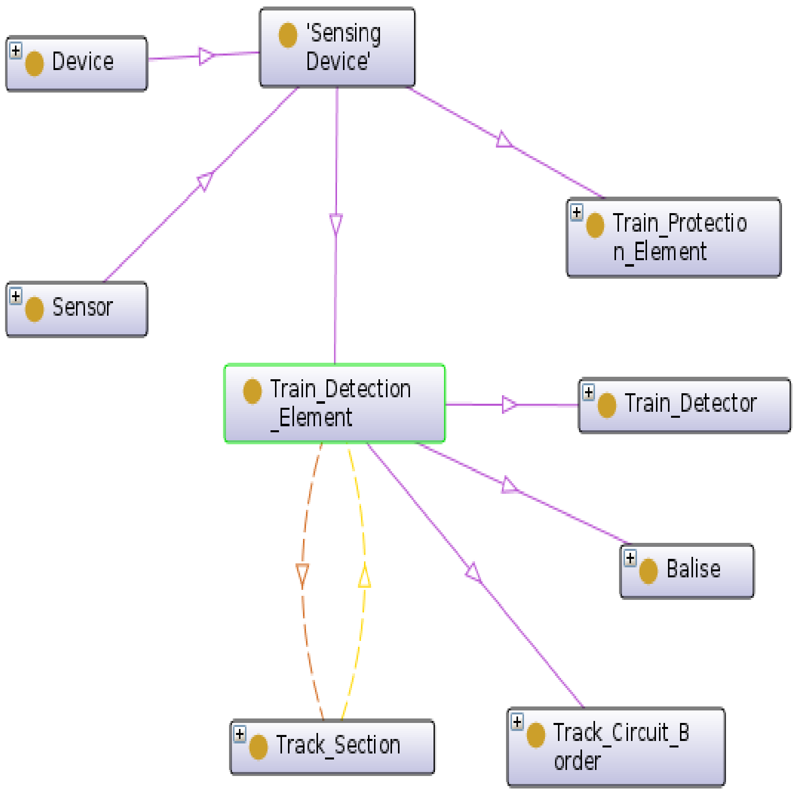


FIG. 8: TRAIN DETECTION ELEMENT AS EXTENSION OF SSN:DEVICE

Before using the ontology it is important to evaluate it in terms of consistency, completeness, and conciseness. Indeed, during the developing, because manually checking is troublesome and prone to introduce new errors, we used an automated - OOPS (OntOlogy Pitfall Scanner) – documented in [16].

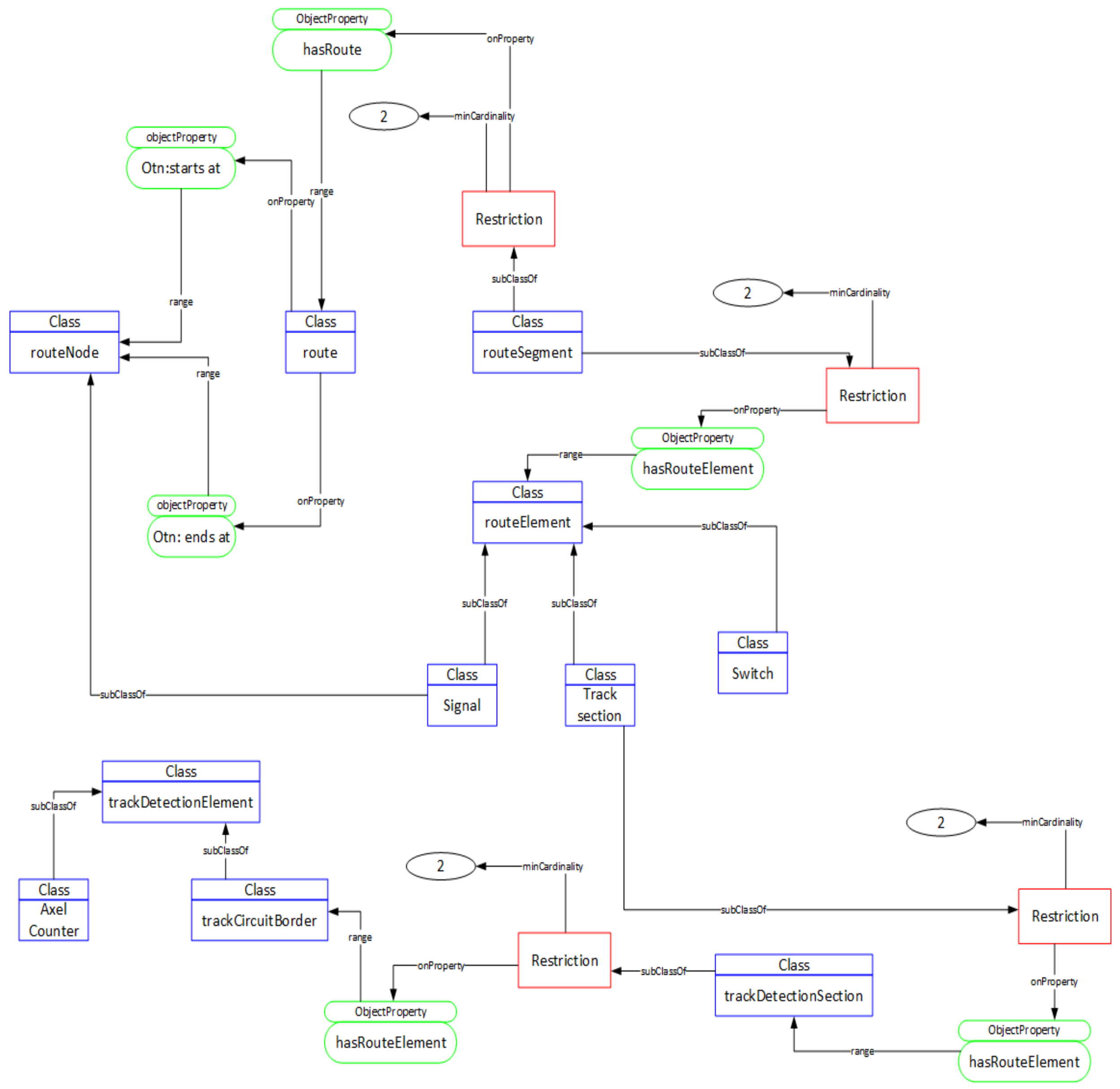


FIG. 9: KBI ONTOLOGY FOCUSING ON ROUTE PART

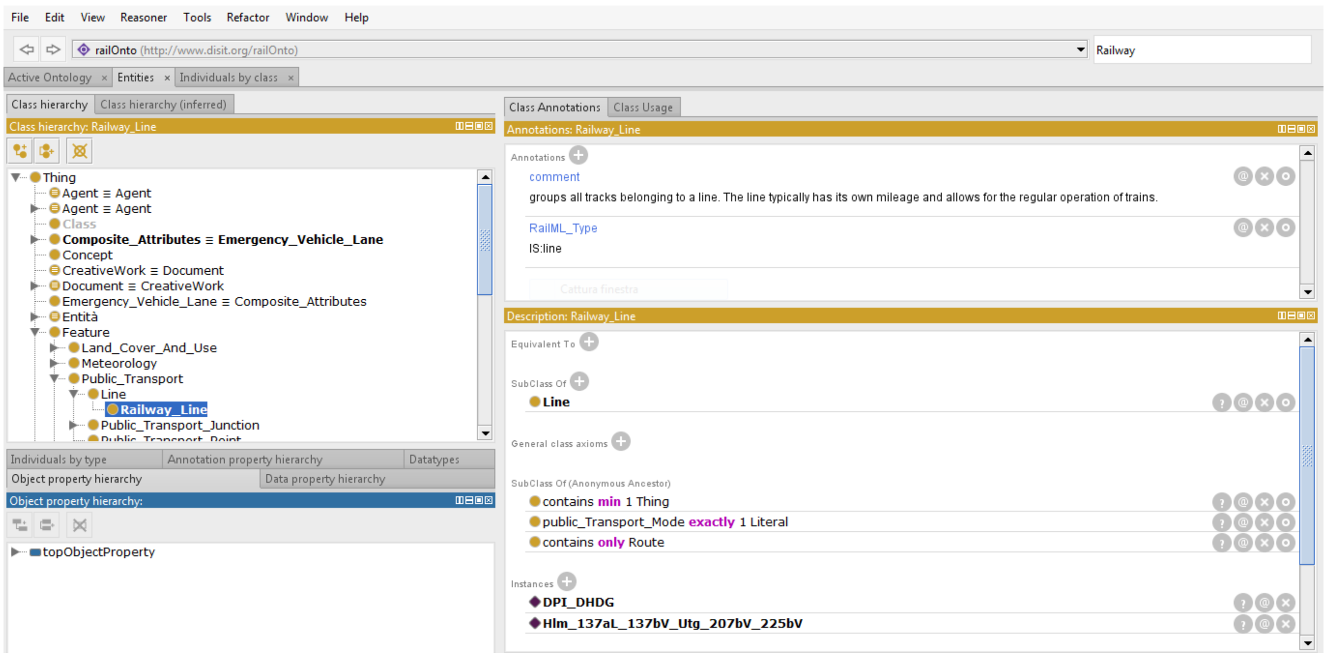
Development environment

In this section we introduce the tools and environment used to edit the ontology and setup the knowledge-base.

Ontology Editor: Protege

Protégé-owl has become the default open-source editor for the Web ontology Language. It was developed collaboratively between University of Manchester and Stanford University based on the earlier Protégé that used the frame-based formalism based on OKBC.

Protégé is an open-source environment that provides a framework for “plugins” either for specific applications or alternative editing tools and views.


File
Edit
View
Reasoner
Tools
Refactor
Window
Help
railOnto (http://www.disit.org/railOnto)
Railway
Active Ontology
Entities
Individuals by class
Class hierarchy
Class hierarchy (inferred)
Class hierarchy: Railway_Line
Class Annotations
Class Usage
Annotations: Railway_Line
Description: Railway_Line
Individuals by type
Annotation property hierarchy
Datatypes
Object property hierarchy
Data property hierarchy
Object property hierarchy:
Cattura finestra

```
Thing
  Agent ≡ Agent
  Agent ≡ Agent
  Class
  Composite_Attributes ≡ Emergency_Vehicle_Lane
  Concept
  CreativeWork ≡ Document
  Document ≡ CreativeWork
  Emergency_Vehicle_Lane ≡ Composite_Attributes
  Entità
  Feature
    Land_Cover_And_Use
    Meteorology
    Public_Transport
      Line
        Railway_Line
      Public_Transport_Junction
      Public_Transport_Point

topObjectProperty
```

```
Annotations
  comment
  groups all tracks belonging to a line. The line typically has its own mileage and allows for the regular operation of trains.
  RailML_Type
  IS:line

Equivalent To

SubClass Of
  Line

General class axioms

SubClass Of (Anonymous Ancestor)
  contains min 1 Thing
  public_Transport_Mode exactly 1 Literal
  contains only Route

Instances
  DPI_DHDG
  Hlm_137aL_137bV_Utg_207bV_225bV
```

FIG. 10: PROTÉGÉ EDITOR

The current Version (4) is a complete re-write including the full OWL 2 specification and built on the new OWL 2 API and a range of plugins for easy creation of OWL ontologies – e.g. outline and spreadsheet like functionality, better search facilities, and tighter integration with OWL reasoner, etc.

RDF-Store: Stardog

Stardog [17] is a lightweight RDF database developed by Clark &Parsia who are the maintainers of Pellet Reasoner.

The store is advertised as being a graph database that uses RDF data, SPARQL for queries, OWL for reasoning, and pure java for the enterprise. "Two existing modes are supported in Stardog, one based only triples and another one for quads. In both cases, the indexes are store on-disk, but in-memory mode is available. No other details are provided in any publication. As a graph database, Stardog supports ACID transactions, SPARQL 1.1, and full text search through the use of Lucene. Among the original features of Stardog, we can highlight the support of Integrity Constraint validation which adopts a closed-word assumption in a declarative and high level manner but enabling integrity constraints to be defined in OWL, SWRL or SPARQL.

Integrity Constraint

Recalling that Web ontology language OWL follows "open world assumption" (OWA) semantics that implies every statement whose truth is not known to be undefined rather than false in the contrasting "closed world assumption" (CWA) semantics.

While OWA semantics is appropriate for many traditional OWL uses, in the recent years there have been also efforts to introduce "integrity constraints" over OWL ontology models through CWA semantics. The integrity constraint (IC) assertions may appear natural in e.g. information system specifications, where, for example, a missing train detector for a track section under the assertion that every track section has a train detector would be naturally interpreted as a data error rather than inferring existence of some unknown train detector for the track section considered.

The IC specification implemented in Stardog OWL/RDF database, reuse the OWL syntax itself also for IC thus materializing the idea of using "the full expressivity of OWL and OWL 2 as a schema language for RDF"1.

There are five type of constraint:

- Subsumption Constraints which guarantees certain subclass and superclass (i.e., Subsumption) relationships exist between instances.
- Domain-Range Constraints which control the types of domain and range instances of properties.
- Participation Constraints which control whether or not an RDF instance participates in some specified relationship
- Cardinality Constraints which control the number of various relationships or property values.
- Property Constraints which control how instances are related to one another via properties.

For example we could validate if a switch is connected to a least two track section:

```
:Switch rdfs:subClassOf

[a owlRestriction;

owl:onProperty :is_connected_to;

owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger

owl:onClass :Track_Section

].
```

Or we could check that a switch must not be connected to more than X track section

```
:Switch rdfs:subClassOf

[ a owl:Restriction ;

owl:onProperty :connected_to;

owl:maxQualifiedCardinality "3"^^xsd:nonNegativeInteger

owl:onClass :Track_Section

] .
```

It is possible also to convert such constraint in direct SPARQL queries. For example to check if a train detector it not is connected to a track section

```
SELECT  *
WHERE {
   ?x1 a :Train_Detector.
   FILTER NOT EXISTS {
       ?x1 :is_train_detector_for ?x0

   }
}
```

Validation

Validation is the process of analyzing the knowledge and decision-making capabilities of the expert system [11]. At least six different possibilities exist for analyzing expert systems:

- Analyze the knowledge base for accuracy,
- Analyze the kb for completeness,
- Analyze the kb weights,
- Test the inference engine,
- Analyze condition-decision matches for decision quality,
- And analyze condition-decision matches to determine whether the right answer was found for the right reasons.

One of a possible work result consisted into generating a control table for a track layout. This entail the disclosure of all possible routes hence we must to be able to detect the route's track section and points in reverse or normal position depending by the direction of movement. Also, the track layout have to be correctly deployed, for example it could be exist an orphan track section or a track section without signals or train detection elements. We individuated two classes of verification cases: one related to routes over track layout and one related to **correctness** of deployed track elements. Each investigation corresponds to a couple related to checking Completeness and Soundness properties.

Routes checking

1. Generic routes
   a. **C**: It is possible to identify all routes?
   b. **S**: All given results are routes?
2. Inverse routes
   a. **C**: It is possible to identify all possible inverse routes?
   b. **S**: All given results are inverse route?
3. Opposite routes
   a. **C**: It is possible to identify all possible opposite routes?
   b. *S*: all given results are opposite routes?

Correctness of deployed elements

1. Correct track section
   a. **C**: It is possible to identify all correct track sections?
   b. **S**: All given results are track sections?
2. Switches
   a. **C**: It is possible to identify all possible switches?
   b. **S**: All given results are switches?

Experimental results

The workbench of our test were three station’s track layout. The first is an example layout with a corresponding control table. The latter two are a part of real stations, respectively a Dutch’s one which is reported in figure 6 and German’s ones which is reported in figure 7.

Santpoor Noord and Banhoff Pirna, also, are representative for an average, medium complex interlocking area as they contains various track conditions/elements:

- 6(8) switches;
- 10(11)signals;
- 2(0) crossovers;
- 21 detection sections;
- 2 speed regimes;
- 18 routes.

Being a track layout planning is related to a national rules, the above choice permitted us to test different environment.

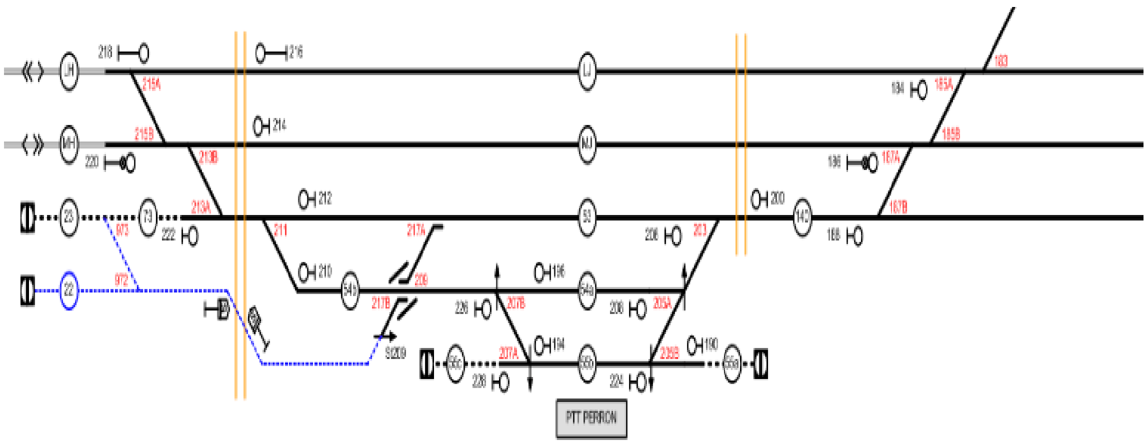


FIG. 11: SANTPOOR NOORD STATION

An example of snippet code of the railml file of Santpoor station is reported

```
<track id="Sptn_1405R_1427L"  description="Sptn station spoor 1 met perron">

<trackTopology>

<trackBegin id="Sptn_1405R"  pos="0" absPos="5.41">

       <connection id="Sptn_1405RV"  ref="Sptn_1405VR" />

</trackBegin>

<trackEnd id="Sptn_1427L"  pos="0.703" absPos="6.113">

       <connection id="Sptn_1427LV"  ref="Sptn_1427VL" />

</trackEnd>

</trackTopology>

<trackElements>

<gradientChanges>

       <gradientChange id="Sptn_1405RX" pos="0"  absPos="5.41" dir="up" slope="-0.221" />
```

```
        <gradientChange id="Sptn_1405R_800"  pos="0.358" absPos="5.768"  dir="up"  slope="0.678" />
        <gradientChange id="Sptn_1405R_1300"  pos="0.653" absPos="6.063" dir="up"  slope="0.056" />
</gradientChanges>
<speedchanges>
        <speedchange id="Sptn_1405R_X1"  pos="0.358" absPos="5.768" vmax="100" dir="down" />
        <speedchange id="Sptn_1405R_X2"  pos="0.653"  absPos="6.063" vmax="100" dir="up" />
</speedchanges>
<levelCrossings>
        <levelCrossing id="Sptn_1405R_1427L_5.76" pos="0.35" absPos="5.760" />
</levelCrossings>
</trackElements>
<ocsElements>
<signals>
        <signal id="Sptn_1412" pos="0.358"  absPos="5.768"  name="1412"  dir="down" >
                <speed>
                        <speedChangeRef ref="Sptn_1405R_X1"/>
                </speed>
        </signal>
        <signal id="Sptn_1426"  pos="0.653" absPos="6.063"  name="1426" dir="up" >
                <speed>
                        <speedChangeRef ref="Sptn_1405R_X2"/>
                </speed>
        </signal>
</signals>
<trainDetectionElements>
        <trackCircuitBorder id="Sptn_1405R_200"  pos="0.094"  name="1404CT_1405T" absPos="5.504" />
        <trackCircuitBorder id="Sptn_1405R_400" name="1404CT_1404DT" absPos="5.736" />
        <trackCircuitBorder id="Sptn_1405R_600"  pos="0.356" name="1404DT_1404ET" absPos="5.766" />
        <trackCircuitBorder id="Sptn_1405R_1400"  pos="0.662"  name="1404ET_1427T" absPos="6.072"/>
        </trainDetectionElements>
</ocsElements>
</track>
```

The relative RDF triples:

```
<!-- http://www.disit.org/railOnto#Sptn_piattaforma_stazione_1 -->
```

```
    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_piattaforma_stazione_1">

        <rdf:type rdf:resource="http://www.disit.org/railOnto#Track_Section"/>

        <track_section_id>Sptn_1405R_1427L</track_section_id>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405RX"/>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_1300"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_1400"/>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_1427L_5.27"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_200"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_400"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_600"/>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_800"/>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_X1"/>

        <has_track_elements rdf:resource="http://www.disit.org/railOnto#Sptn_1405R_X2"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1412"/>

        <ssn:attachedSystem rdf:resource="http://www.disit.org/railOnto#Sptn_1426"/>

        <OTN:starts_at
rdf:resource="http://www.disit.org/railOnto#Sptn_station_spoor_1_met_perron_TB"/>

        <OTN:ends_at rdf:resource="http://www.disit.org/railOnto#Sptn_station_spoor_1_met_perron_TE"/>

    </owl:NamedIndividual>

   <!-- http://www.disit.org/railOnto#Sptn_1405RX -->

       <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405RX">
           <rdf:type rdf:resource="http://www.disit.org/railOnto#Gradient_Change"/>
           <gradient_change_slope rdf:datatype="&xsd;float">-0.221</gradient_change_slope>
           <gradient_change_pos rdf:datatype="&xsd;float">0.0</gradient_change_pos>
           <gradient_change_abspos rdf:datatype="&xsd;float">5.41</gradient_change_abspos>
           <gradient_change_dir>up</gradient_change_dir>
           <gradient_change_id>Sptn_1405RX</gradient_change_id>
       </owl:NamedIndividual>

   <!-- http://www.disit.org/railOnto#Sptn_1405R_1300 -->

       <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_1300">
           <rdf:type rdf:resource="http://www.disit.org/railOnto#Gradient_Change"/>
           <gradient_change_slope rdf:datatype="&xsd;float">0.056</gradient_change_slope>
           <gradient_change_pos rdf:datatype="&xsd;float">0.653</gradient_change_pos>
           <gradient_change_abspos rdf:datatype="&xsd;float">6.063</gradient_change_abspos>
           <gradient_change_id>Sptn_1405R_1300</gradient_change_id>
           <gradient_change_dir>up</gradient_change_dir>
       </owl:NamedIndividual>


<!-- http://www.disit.org/railOnto#Sptn_1405R_1400 -->

 <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_1400">

        <rdf:type rdf:resource="http://www.disit.org/railOnto#Track_Circuit_Border"/>

        <track_circuit_border_pos rdf:datatype="&xsd;float">0.662</track_circuit_border_pos>

        <track_circuit_border_abspos rdf:datatype="&xsd;float">6.072</track_circuit_border_abspos>

        <track_circuit_border_name>1404ET_1427T</track_circuit_border_name>

        <track_circuit_border_id>Sptn_1405R_1400</track_circuit_border_id>

    </owl:NamedIndividual>
```

```xml
<!-- http://www.disit.org/railOnto#Sptn_1405R_1427L_5.27 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_1427L_5.27">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Level_Crossing"/>
        <level_crossing_pos rdf:datatype="&xsd;float">0.35</level_crossing_pos>
        <level_crossing_abspos rdf:datatype="&xsd;float">5.76</level_crossing_abspos>
        <level_crossing_id>Sptn_140tR_1427L_5.27</level_crossing_id>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1405R_200 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_200">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Track_Circuit_Border"/>
        <track_circuit_border_pos rdf:datatype="&xsd;float">0.094</track_circuit_border_pos>
        <track_circuit_border_abspos rdf:datatype="&xsd;float">5.504</track_circuit_border_abspos>
        <track_circuit_border_name>1404CT_1405T</track_circuit_border_name>
        <track_circuit_border_id>Sptn_1405R_200</track_circuit_border_id>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1405R_400 -->
    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_400">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Track_Circuit_Border"/>
        <track_circuit_border_pos rdf:datatype="&xsd;float">0.326</track_circuit_border_pos>
        <track_circuit_border_abspos rdf:datatype="&xsd;float">5.736</track_circuit_border_abspos>
        <track_circuit_border_name>1404CT_1404DT</track_circuit_border_name>
        <track_circuit_border_id>Sptn_1405R_400</track_circuit_border_id>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1405R_600 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_600">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Track_Circuit_Border"/>
        <track_circuit_border_pos rdf:datatype="&xsd;float">0.356</track_circuit_border_pos>
        <track_circuit_border_abspos rdf:datatype="&xsd;float">5.766</track_circuit_border_abspos>
        <track_circuit_border_id>Sptn_1405R_600</track_circuit_border_id>
        <track_circuit_border_name>1404DT_1404ET</track_circuit_border_name>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1405R_800 -->
```

```
    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_800">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Gradient_Change"/>
        <gradient_change_pos rdf:datatype="&xsd;float">0.358</gradient_change_pos>
        <gradient_change_slope rdf:datatype="&xsd;float">0.678</gradient_change_slope>
        <gradient_change_abspos rdf:datatype="&xsd;float">5.768</gradient_change_abspos>
        <gradient_change_dir>up</gradient_change_dir>
        <gradient_change_id>Sptn_1405R_800</gradient_change_id>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1405R_X1 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_X1">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Speed_Changes"/>
        <speed_change_pos rdf:datatype="&xsd;float">0.358</speed_change_pos>
        <vmax rdf:datatype="&xsd;float">100.0</vmax>
        <speed_change_abspos rdf:datatype="&xsd;float">5.768</speed_change_abspos>
        <speed_change_dir>down</speed_change_dir>
        <speed_change_id>Sptn_1405R_X1</speed_change_id>
    </owl:NamedIndividual>



    <!-- http://www.disit.org/railOnto#Sptn_1405R_X2 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1405R_X2">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Speed_Changes"/>
        <speed_change_pos rdf:datatype="&xsd;float">0.653</speed_change_pos>
        <vmax rdf:datatype="&xsd;float">100.0</vmax>
        <speed_change_abspos rdf:datatype="&xsd;float">6.063</speed_change_abspos>
        <speed_change_id>Sptn_1405R_X2</speed_change_id>
        <speed_change_dir>up</speed_change_dir>
    </owl:NamedIndividual>

<!-- http://www.disit.org/railOnto#Sptn_1412 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1412">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Signal"/>
        <signal_pos rdf:datatype="&xsd;float">0.358</signal_pos>
        <signal_name rdf:datatype="&xsd;string">1412</signal_name>
        <signal_abspos rdf:datatype="&xsd;float">5.768</signal_abspos>
        <signal_id>Sptn_1412</signal_id>
```

```
     <signal_dir>up</signal_dir>

 </owl:NamedIndividual>


<!-- http://www.disit.org/railOnto#Sptn_1426 -->

    <owl:NamedIndividual rdf:about="http://www.disit.org/railOnto#Sptn_1426">
        <rdf:type rdf:resource="http://www.disit.org/railOnto#Signal"/>
        <signal_pos rdf:datatype="&xsd;float">0.653</signal_pos>
        <signal_name rdf:datatype="&xsd;string">1426</signal_name>
        <signal_abspos rdf:datatype="&xsd;float">6.0623</signal_abspos>
        <signal_id>Sptn_1426</signal_id>
        <signal_dir>up</signal_dir>
    </owl:NamedIndividual>
```

Following a test conducted on validating the stations proving consistency and completeness

```
Stardog server is listening on all network interfaces.
SNARL server available at snarl://localhost:5820.
HTTP server available at http://localhost:5820.

STARDOG_HOME=/home/zaza/stardog-2.2.3

LOG_FILE=/home/zaza/stardog-2.2.3/stardog.log
Cleanin previosoly DB
Successfully deleted database 'RailOntoDB'.
Creating db in stardog with railOntoNuova-TBOX.owl as Ontology
Bulk loading data to new database.

Parsing triples: 0% complete in 00:00:00 (0 triples - 0.0K triples/sec)
Parsing triples: 100% complete in 00:00:00 (1.2K triples - 1.9K triples/sec)
Parsing triples finished in 00:00:00.622

Creating index: 0% complete in 00:00:00 (0.0K triples/sec)
Creating index: 100% complete in 00:00:00 (13.3K triples/sec)
Creating index finished in 00:00:00.089

Computing statistics: 0% complete in 00:00:00 (0.0K triples/sec)
Computing statistics: 100% complete in 00:00:00 (10.1K triples/sec)
Computing statistics finished in 00:00:00.118
Loading complete.
Inserted 1,186 triples in 00:00:01.066 at 1.1K triples/sec
Bulk load complete.  Loaded 1,186 triples from 1 file(s) in 00:00:01 @ 1.2K triples/sec.

Successfully created database 'RailOntoDB'.

START TESTING
Adding Complex Constraints
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix : <http://www.disit.org/railOnto#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/'> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


#All Track section must have exactly one Train Detector or any subclass of it.


#Terp Syntax



#Turtle Syntax


:Track_Section rdfs:subClassOf
    [ a owl:Restriction ;

    owl:onProperty ssn:attachedSystem ;
```

```
        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;

                    owl:onClass :Train_Detection_Element
        ] .
        Successfully added constraints in 00:00:00.

    CONVERTING in SPARQL queries

    Removing Comments
    #Constraintss: 1
    #Constraints:   AxiomConstraint{:Track_Section   rdfs:subClassOf   (ssn:attachedSystem   exactly   1
:Train_Detection_Element)}
    #Query:
    SELECT DISTINCT *
    FROM <tag:stardog:api:context:all>
    WHERE {
       ?x0                                         <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.disit.org/railOnto#Track_Section> .
       {
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#attachedSystem> ?x1 .
          ?x1                                      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.disit.org/railOnto#Train_Detection_Element> .
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#attachedSystem> ?x2 .
          ?x2                                      <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.disit.org/railOnto#Train_Detection_Element> .
          FILTER (?x1 != ?x2)
       }
       UNION
       {
          FILTER NOT EXISTS {
             ?x0 <http://purl.oclc.org/NET/ssnx/ssn#attachedSystem> ?x3 .
             ?x3                                   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.disit.org/railOnto#Train_Detection_Element> .
          }
          ?x0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> .
       }
    }

    @prefix owl: <http://www.w3.org/2002/07/owl#> .
    @prefix : <http://www.disit.org/railOnto#> .
    @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
    @prefix foaf: <http://xmlns.com/foaf/0.1/'> .
    @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
    @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


    #There must be at least one Track_Section with an Home Signal(Entry)

    #Terp Syntax


        # :Home_Signal rdfs:subClassOf (ssn:onPlatform exactly 1 :Track_Section).


    :Home_Signal rdfs:subClassOf
        [ a owl:Restriction ;

        owl:onProperty ssn:onPlatform ;

        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;

                    owl:onClass :Track_Section
        ]


    .
        Successfully added constraints in 00:00:00.
    CONVERTING in SPARQL queries
    Removing Comments
    #Constraintss: 1
    #Constraints:    AxiomConstraint{:Home_Signal    rdfs:subClassOf    (ssn:onPlatform    exactly    1
:Track_Section)}
    #Query:
    SELECT DISTINCT *
    FROM <tag:stardog:api:context:all>
    WHERE {
       ?x0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.disit.org/railOnto#Home_Signal>
 .
       {
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x1 .
          ?x1                                             <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x2 .
          ?x2                                             <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
          FILTER (?x1 != ?x2)
       }
       UNION
       {
          FILTER NOT EXISTS {
             ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x3 .
             ?x3                                          <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
          }
          ?x0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> .
       }
    }

    @prefix owl: <http://www.w3.org/2002/07/owl#> .
    @prefix : <http://www.disit.org/railOnto#> .
    @prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
    @prefix foaf: <http://xmlns.com/foaf/0.1/'> .
    @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
    @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


    #There must be at least one Track_Section with an Starter_Signal (Exit)

    #Terp Syntax


    # :Starter_Signal rdfs:subClassOf (ssn:onPlatform exactly 1 :Track_Section).


    :Starter_Signal rdfs:subClassOf
        [ a owl:Restriction ;

        owl:onProperty ssn:onPlatform ;

        owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;

                    owl:onClass :Track_Section
        ]


    .
        Successfully added constraints in 00:00:00.
    CONVERTING in SPARQL queries
    Removing Comments
    #Constraintss: 1
    #Constraints:    AxiomConstraint{:Starter_Signal    rdfs:subClassOf    (ssn:onPlatform    exactly    1
 :Track_Section)}
    #Query:
    SELECT DISTINCT *
    FROM <tag:stardog:api:context:all>
    WHERE {
       ?x0                                                <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Starter_Signal> .
       {
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x1 .
          ?x1                                             <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
          ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x2 .
          ?x2                                             <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
          FILTER (?x1 != ?x2)
       }
       UNION
       {
          FILTER NOT EXISTS {
             ?x0 <http://purl.oclc.org/NET/ssnx/ssn#onPlatform> ?x3 .
             ?x3                                          <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://www.disit.org/railOnto#Track_Section> .
```

```
        }
        ?x0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2002/07/owl#Thing> .
      }
    }
```

```
   Adding Data: Simple Station Correct
   Adding      data      from      file:      /home/zaza/modello/ontology/A-Box/railOntoNuova-ABOX-
   SimplePassThroughStation.owl
   Added 94 triples in 00:00:00.217
   Validating with Reasoner=QL
   Data is valid.
   Query using the reasoner
   +--------+--------+--------+--------+
   |  x0    |  x1    |  x2    |  x3    |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+

   Query returned 0 results in 00:00:03.994
   +--------+--------+--------+--------+
   |  x0    |  x1    |  x2    |  x3    |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+

   Query returned 0 results in 00:00:00.256
   +--------+--------+--------+--------+
   |  x0    |  x1    |  x2    |  x3    |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+

   Query returned 0 results in 00:00:00.411
   Removing Data
   Removing      data      from      file:      /home/zaza/modello/ontology/A-Box/railOntoNuova-ABOX-
SimplePassThroughStation.owl
   Removed 99 triples in 00:00:00.224

   Adding Data: Simple Station Not Correct because not all track sections are monitored by a train
detection elements (Balise, Axel Counter or Track Circuit

   Adding      data      from      file:      /home/zaza/modello/ontology/A-Box/railOntoNuova-ABOX-
SimplePassThroughStation-ERROR_Train_Detectors.owl
   Added 99 triples in 00:00:00.215
   Validating with Reasoner=QL
   Data is NOT valid.
   The following constraints were violated:
   AxiomConstraint{:Track_Section      rdfs:subClassOf      (ssn:attachedSystem      exactly      1
:Train_Detection_Element)}
   Query using the reasoner
   +-----------------------------------------+-----------------------------------------+----------------
------------------------+--------+
   |                       x0                        |                       x1                        |
x2                  |  x3    |
   +-----------------------------------------+-----------------------------------------+----------------
------------------------+--------+
   |     :samplePassThroughStationTrackSectionT1     |     :samplePassThroughStationTrackCircuit1     |
:samplePassThroughStationTrackCircuit2 |        |
   |     :samplePassThroughStationTrackSectionT1     |     :samplePassThroughStationTrackCircuit2     |
:samplePassThroughStationTrackCircuit1 |        |
   |  :samplePassThroughStationTrackSectionT2  |                                                 |
|        |
   +-----------------------------------------+-----------------------------------------+----------------
------------------------+--------+

   Query returned 3 results in 00:00:00.610
   +--------+--------+--------+--------+
   |  x0    |  x1    |  x2    |  x3    |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+
   Query returned 0 results in 00:00:00.191
   +-------+-------+-------+-------+
   |  x0   |  x1   |  x2   |  x3   |
   +-------+-------+-------+-------+
   +-------+-------+-------+-------+

   Query returned 0 results in 00:00:00.276
```

As we mention in the sections above our study is conducted translating the competency questions into SPARQL queries.

For example, related to identifying or not a correct track section in the stored track layout, we search for elements that satisfy the property of being a track section in the RDF-Store and we check if all results are effective a track sections. This is realized in the listing below.

The test conducted fulfilled our established assumption, also, we was able to generating a simplified control table (see I) to the real cases by translating the query results.

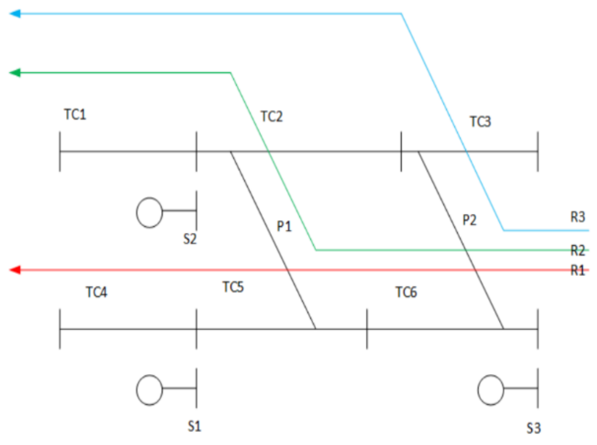


FIG. 12: SIMPLE LAYOUT STATION

TABLE I. CONTROL TABLE RELATED TO EXAMPLE LAYOUT

Route	***Tracks***	***Point normal***	***Point Reverse***
1(S2-S1)	TC4, TC5, TC6	P1, P2	-
2(S2-S3)	TC6, TC2, TC1	P2	P1
3(S2-S3)	TC3, TC2, TC1	P1	P2

Bibliography

[1] International Union of Railways, "http://www.uic.org/".

[2] European Committe for Electrotechnical Standarization, "http://www.cenelec.eu/".

[3] Intelligent integration of railway systems, http://www.integrail.info/.

[4] P. Umiliacchi, U. Henning, G. Langer and R. Shingler, "Standardized Data Interchange Between Railway Systems: an Integrated Railway Information System".

[5] S. Verstichel, F. Ongenae, L. Loeve, F. Vermeulen, P. Dings, B. Dhoedt, T. Dhaene and F. De Turck, "Efficient data integration in the railway domain through an ontology-based methodology," *Transportation Research Part C: Emerging Technologies,* vol. 19, no. 4, pp. 617-643, 2011.

[6] M. Lodemann and N. Luttenberger, "Ontology-based Railway Infrastructure Verification-Planning Benefits.," in *KMIS*, 2010.

[7] M. Lodemann, N. Luttenberger and E. Schulz, "Semantic Computing for Railway Infrastructure Verification," in *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, 2013.

[8] D. Briola, R. Caccia, M. Bozzano and A. Locoro, "Ontologica: Exploiting ontologies and natural language for railway management. Design, implementation and usage examples," *International Journal of Knowledge-based and Intelligent Engineering Systems,* vol. 17, no. 1, pp. 3-15, 2013.

[9] A. Nash, D. Huerlimann, J. Sch{\"u}tte and V. P. Krauss, "RailML-a standard data interface for railroad applications," *Publication of: WIT Press,* 2004.

[10] A. Barr, E. Feigenbaum and P. Cohen, The Handbook of artificial intelligence, HeurisTech Press, 1982.

[11] J. C. Giarratano and G. Riley, Expert Systems: Principles and Programming, Pacific Grove, CA, USA: Brooks/Cole Publishing Co., 1989.

[12] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition,* vol. 5, no. 2, pp. 199-220, 1993.

[13] M. E. Foster and M. White, "Techniques for text planning with {XSLT}," in *Proceedings of the 4th Workshop on NLP and XML (NLPXML 2004)*, Barcelona, 2004.

[14] Rewerse Project, "Rewerse Project," [Online]. Available: http://rewerse.net.

[15] M. Compton, P. Barnaghi, L. Bermudez, R. Garc{\'\i}A-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog and others, "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 17, pp. 25-32, 2012.

[16] M. Poveda Villalon and M. C. Su{\'a}rez-Figueroa, "OOPS!--OntOlogy Pitfalls Scanner!," 2012.

[17] Clark&Parsia, "Stardog Enterprise Graph Database," [Online]. Available: http://www.stardog.com.

[18] E. Sirin and J. Tao, "Towards Integrity Constraints in OWL.," in *OWLED*, 2009.

[19] C. Li and T. W. Ling, "From XML to semantic web," in *Database Systems for Advanced Applications*, 2005.