UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Dipartimento di Ingegneria dell'Informazione (DINFO)
Corso di Dottorato in Ingegneria dell'Informazione
Curriculum:Telematica e Società dell'Informazione

Ontological Knowledge-base for railway control system and Analytical Data Platform for Twitter

Settore Scientifico Disciplinare ING-INF/05

Candidate
Imad Zaza

Supervisors
Prof. Paolo Nesi

Prof. Alessandro Fantechi

PhD Coordinator
Prof.Luigi Chisci

ciclo XXX, 2014-2017

Università degli Studi di Firenze, Dipartimento di Ingegneria dell'Informazione (DINFO).

A mio figlio Filippo

Acknowledgements

I would like to acknowledge the efforts and input of my supervisor, Prof. Paolo Nesi, and all my colleagues of the DISIT Lab (DISIT) who were of great help during my research.

In particular my thanks go to Pierfrancesco Bellini, Claudio Badii, Irene Paoli, Stefano Bilotta, Gianni Pantaleo, Angelo Difino, Michela Paolucci, Nicola Mitolo, Daniele Cenni, Lorenzo Massai, Mirko Soderi, Mino Marazzini, Daniele Bologna, Emanuele Bellini and Simonetta Ceglia.

A special thank to Prof. Alessandro Fantechi.

I would like to thank Martina my beloved wife for all this time together; also Grazia, Filippo and Milena.

A special thank to my little son Filippo, which is a continuous source of power.

I would always be grateful to my family: my Father Rami Zaza and my Mother Laudi Lteif, my Brothers Jad and Shadi and their families

I am grateful to Prof. Pietro Zecca and Prof. Giuseppe Anichini for time spent together during these years.

A special mention to Prof. Giovanni Frosali, Prof. Giuseppe Modica, Prof. Paolo Maria Mariano and Prof. Mario Landucci who always pushed to keep me going on.

I would like to thank my Tesys Rails course coordinator Susanna Papini and colleagues Matteo, Simone, Amedeo, Benedetta, Martina and Elisa.

I cannot omit to thank my friends Massimiliano Masi, Leonardo - Guitar - Passigli, Gabriele Galletti, Lorenzo - King - Brugnoni, Leonardo - man in the woods - Donati, Andrea Barducci and Alberto - Il Conte - Piani.

Finally my thoughts goes to my dear cousin Rabeah Lteif: we miss you.

I am ironman

Contents

Introduction

The objective

Railway signalling

Railway signalling is a set of standards and physical devices (light signals, deviations, train detectors, etc.) ensuring the safety of train traffic on railway infrastructure.

Railway Interlocking System (IXL) implements the commands to the aforesaid devices in compliance with the requirements for requests received from the traffic control system. If these requests do not violate safety then the signals are properly committed on the physical infrastructure. The computational model of these systems is equivalent to a parallel signal processor. More specifically, these systems are based on a table of conditions which contains for each itinerary the status that the device must have. This table is compiled for each route by a railway engineer, so it is unique for each track and reflects the different operating standards per nation.

The aim of the research activity consisted in the study and development of an IXL ontological model. In this regard, the activity concerned:

- The study of interlocking systems, including literature review, standards governing them and software tools for configuring them;
- Static development, verification and validation of an abstract static model of railway infrastructure.

The works considered relevant were those that fully formalized the topology of a railway network (static and dynamic), thus showing the consistency and completeness of the proposed method. During this phase, some inter-

locking principles have been extrapolated to railway operators from different countries (Denmark, Sweden, Holland, Germany).

The paradigm adopted is the ontological one, and in particular, ontological representations based on the semantic web and the reasoning tools on the safety of the railway track configuration. As this innovative approach, it was necessary to study and then to provide first of all the evidence of correctness, validity of the reasoning and expressibility tools of the representation system.

Social media analysis

This research activity was carried out using the Twitter Vigilance tool produced by the DISIT Lab.

Social network analysis has undergone a renaissance with the ubiquity and quantity of content from social media, web pages, and sensors. This content is a rich data source for analysing human sensors and social networks, but its enormity and unstructured nature also present multiple challenges. Work at DISIT Lab is addressing the problems developing tool to tackle this problem.

Twitter Vigilance is a multi-user platform on-the-edge that allows you to do Data-Analytic and Sentiment-Analysis on Twitter Social Media. This translates into the ability to control and analyse the level of appreciation and / or dissent (for example, an event, product or people) or to conduct short- and long-term comparisons of results, nearly real-time identifications the emergence of explosive events, critical situations etc. Twitter Vigilance is part of the echo system km4city, the back-bone of the H22020Si-mobility and resolute projects that allows interconnection, storage and subsequent heterogeneous data from various institutional realities (for example: portals in the region Tuscany like MIIC, Moving to Tuscany, Traffic Observatory), Open-Data provided by individual municipalities. In this regard, among the experiments currently conducted on the Twitter Vigilance platform, that on the analysis of perceived service quality, it provides an interesting methodology for assessing the efficiency of Local Public Transport (TPL), expressed as the relationship between quality of service desired and quality of service perceived.

Twitter Vigilance is a Big Data platform because the tweets acquisition rate is high (for example, the data collected from April 2015 is now in the order of 300 million lines) with a high degree of variation (in fact the same

tweet it is updated because it could be re-deployed or have an add-on). This aspect has led to studying the reformulation of the single host storage architecture into a distributed data storage architecture. Big Table introduced by Google Inc. in 2005 it is a distributed storage system that is structured as a large table whose dimensions are in the order of petabytes that are distributed among tens of thousands of machines. Big Tables along with NoSQL language are the natural structure and paradigm for managing Big-Data.

The Map-Reduce framework is a programming template to handle a large set of data across a distributed and parallel algorithm on a cluster. For top notes, a Map-Reduce program consists of two procedures: a Map () procedure that performs filtering and sorting and a Reduce () procedure that performs summation operations. The novelty of the framework is not in the former division that starts from functional programming, but introduces scalability and fault tolerant benefits resulting from deployed distributed architecture. All the possible distributed architectures that implement and manage Big-Data have been reviewed. In particular, Apache consortium open-source solutions such as Hadoop, HBase and Oracle have been investigated. The implementation considered interesting is that of the Apache stack.

Contributions

Railway signalling

To tackle this issue an ontology-based railway model is presented. Investigation consisted in formalizing logical rules, which allow us checking completeness, consistency of the model by automatically inferring the layout of the remaining installations i.e. (track circuit, signals and track sections) and the possible train routes through the network. A case study regarding the analysis of two different stations is shown. In particular, the validity of the use of ontological engineering techniques applied to static station configuration has been formally demonstrated.

Social media analysis

The study shows evidence of correlation between twitter data and human sensing by discussing results obtained by using the Twitter Vigilance tool and discusses the problem of porting a single host application on distributed

architecture in the context on twitter data analytic highlighting possible pitfalls.

Part I

Railway signalling: Safety and Security

Part I - Introduction

From the first rail road Liverpool-Manchester in 1830, railway transportation have permitted movement of stock and person, maximizing safety and minimizing pollution, if compared with tire-based transportation or avionic transportation.

The train movement has common and odds if compared with the movement of a car. For example, each trains movement is regulated by rules, and those railway rules have to be strictly respected. But unfortunately, if something is on the tracks, the train cannot avoid it, unlike what happens with a car when an obstacle appears on the street. The long braking distance is due to trains low friction coefficients: during the braking action there are limits to the force that can be transferred to the track. The result is that trains have longer distances. For the sake of clarity, if the velocity is 140 km/h, a braking from 1000 meters before the full stop is normal: this distance is termed the "brake distance". Therefore trains must have at least a brake distance from each other. But looking ahead for 1000 meters or during foggy days is quite a challenge so to tell a train driver that tracks ahead is free, signal are placed next to the track. These signals tell the driver how fast he may go to run his train safe and to ensure he can stop on time.

Normally, signals are placed next to the rail track. These signals help the driver to know the maximum speed that the train can reach to ensure a safety "brake distance".

In the early days of railway signalling, railway administrations have used timetables and flag officers to safeguard the running of trains. However, the increase of the tracks within the railway network brought to a growing number of disasters principally due to human errors. For this reason, automatic signalling equipment have become essential, with those installations, safety is ensured by creating interdependences between the equipment for operation

of the points and the equipment for the operation of the signals [102]. These interdependences are called interlocked, from which the name "interlocking system".

As previously mentioned, the goal of an interlocking system is to secure the travel of a train. This objective is achieved by creating a complex circuit where each component is translated into a logic condition, e.g. "if there is a train at the station, entry signal shall be red" (this condition requires several components though). For example, there is one type of interlocking system covers the open track between stations, while another interlocking system that covers the station itself. Note that, the systems at the stations are by far the most complicated, and for this reason the most interests to study.

This work presents a novelty in modelling a Railway Interlocking System by means of ontology engineering. An interlocking system is a quite extensive system and thus a complete description is considered outside the scope of this work. For this reason only the most essential part of the interlocking system has been modelled. Basically the problem we deal with consists in processing some data in input deriving some new information out of it.

Chapter 1

Railway Interlocking System

This chapter gives required knowledge on railway signalling domain context. In particular, section 1.1 gives a brief history of evolving safety in railways, section 1.2 presents the railway interlocking system and section 1.3 gives the literature review of verification and validation of railway interlocking systems.

1.1 Railway signalling

In the early days of railways transportation there was no system which the train driver could interact with for knowing the state of the line ahead. The driver have to use his eyes, for example, to get sign of a train in front so could stop before hitting it.

Soon, the railway authorities concluded that the driver's sighting distance was not enough to prevent accidents due to human or environment factors as bad brakes or tenuous contact which exists between steel wheel and steel rail for traction and braking. Moreover, the adhesion levels are much lower and vehicle weights much higher on railways than on roads and therefore trains need a much greater distance to stop than, say, a car travelling at the same speed.

The next refinement to improve the train driver's stopping distance was to require time intervals between trains (e.g. 10 minutes), divide the track

into sections requiring that only one train was allowed in one section at one time and coloured flags (red, yellow, green) were used by signal officers to tell drivers how to proceed. In such protocol, a red flag was shown by a signal officer for the first five minutes after a train had departed. If a train arrived after 5 minutes, a Yellow caution signal was shown to the driver. The full-speed green signal was only shown after the full 10 minutes had elapsed.

The above system introduced was neither safe nor efficient because it didn't consider that a train could broke between stations, it require that the trains had same speed and the time interval restricted the number of trains which could run per hour over a given line. As the railways authority needed to run more trains, they gradually began to reduce the time between trains. As they reduced the time, or "headway", the number of trains per hour increased. At the same time too, the number of accidents increased.

A further improvement was obtained by introducing fixed signalling where each section (or block as it is often called) is protected by a fixed signal placed at its entrance for display to the driver of an approaching train. If the section is clear, e.g. there is no train in it, the signal will show a "Proceed" indication. The fixed signal it was usually a raised semaphore arm. The above "Proceed" indication it is usually a green light or "aspect". If, however, the section is occupied by a train, the signal will show a "Stop" indication, usually a red aspect. The next train will be made to wait until the train in front has cleared the section. This is the basis upon which all signalling systems are designed and operated.

Mechanical signals were the first that appeared with box with levers - called signal box - controlling remote signals and points. Originally, the passage of each train through a section was tracked visually by the signal officer. When the train had cleared his section, the signal officer told the signal box on the approach side that his section was now clear and that he could, if required, "accept" another train. The messages between signal boxes were transmitted by a system of bell codes using the electric telegraph.

Another improvement was to use double signals. The approach to use a basic stop/go signal used to protect each section of the line was right as long as the driver of an approaching train was able to see the signal in time to stop but, as it was stated above the sight of a driver is not always reactive, so "distant" signals was provided. Distant signals were placed in such a position that the driver could stop in time if the next stop signal was at danger. Positioning depended on the visibility, curvature, maximum

permitted line speed and a calculation of the trains ability to stop. Originally, distant signals were semaphores, like the stop signals mentioned above. They showed a green light at night if their related stop signal was also green (or clear) and yellow if the stop signal was at red. The red-yellow-green pattern was adopted for colour light signals and eventually used to provide a more sophisticated form of train control.

Another safety feature was introduced in the mid-19th Century: mechanical interlocking of points and signals. The purpose was to prevent the route for a train being set up and its protecting signal cleared if there was already another, conflicting route set up and the protecting signal for that route cleared. The interlocking was performed by a series of mechanically interacting rods connected to the signal operating levers in the signal box. The arrangement of the rods physically prevented conflicting moves being set up. As the systems developed, some larger signal cabins at complex junctions had huge frames of interlocking levers, which gave the name "lever frame" to the row of operating levers in a signal box.

Signal overlaps are calculated to allow for the safe braking distance of the trains using a route. Of course, lengths vary according to the site; gradient, maximum train speed and train brake capacity are all used in the calculation. In the UK, it is the practice to provide a 200 yard (185 m) overlap beyond each main line signal in a colour light installation. Back in 1972 when it was decided upon, it was, after a review of many instances where trains had overrun stop signals, considered the maximum normally required. It was a rather crude risk analysis but it was the best they could afford. In the US, the overlap is considered so important that a whole block is provided as the overlap. It is referred to as "absolute block". This means that there is always a full, vacant block between trains. It's rather wasteful of space and it reduces capacity but it saves the need to calculate and then build in overlaps for each signal, so it's cheaper.

Eventually, by the time signal levers were being replaced by small (miniature) levers or push buttons, mechanical interlocking frames were superseded by relay interlocking. Electro-magnetic relays were used in series to ensure the safety of route setting at junctions. Complex "control tables" were drawn up to design the way in which these relays would interact and to ensure safety and integrity.

In response to the continuing need to reduce risks created by train drivers failing to respond to signal instructions, various forms of driver warning

devices and signal command enforcement systems have been developed.

ATC/ATC are all the automated systems that protect the driver and then the train from possible over-speed or exceed of stop signals. Over the years the various European countries have realized different ATP/ATC systems that have been developed according to the own national requirements and technical standards, and operated according to the own operating rules. This has led to the spreading of incompatible train protection and control systems impeding the cross-border operation of European railway traffic. There were also other technical differences among the European countries relating to rail gauge, electricity voltage, rolling stock design, etc. For example, in Italy the BACC system based on conventional coded track circuits employed at two different carrier frequencies, was used. The goal was to manage two train classes, those operating either above or below 180 km/h [4].

Establishing common rules for the free movement of trains in all European countries became essential with the advent of European integration and with this remarkable diversity of European ATC systems. Following the decision taken by the European Transport minister in December 1989, the EU embarked upon a project to analyse the problems relating to signalling and train control [1] [2]. At the end of 1990, the ERRI (European Institute of Railway Research) began to think to develop a common interoperable ATC/ATC system, which could be adopted in all European countries. Implementing a common interoperable platform for railways and signalling systems to ensure the interoperability of the European rail network's was the final goal of the International standard programme ERTMS. The main objectives of interoperability are based on the need to simplify, improve and develop international railway transport services, contribute towards gradually creating an open and competitive domestic market for the supply of railway systems and construction, renewal, restructuring and operative services, and establish standardised European procedures for assessing conformity with interoperability requirements [1]. To define the TSI (Technical Specification for Interoperability) a group of railway experts called ERTMS Group was created, consisting originally of DB, FS and SNCF, but later joined by other railway European companies. To finalize the TSI in the summer of 1998 the UNISIG union was formed, comprising the European Signalling companies Alcatel, Alstom, Ansaldo Signal, Bombardier, Invensys Rail and Siemens. The standardized, interoperable command and control ATC/ATC system was called ERTMS/ETCS, or simply ETCS. Thanks to this system, from

the beginning of the 21st century interoperability of the European rail network is going to be guaranteed. ETCS is divided into different functional levels as established by the set of specifications Baseline 3 (SRS 3.3.0) [3]. The definition of the levels depends on how the rail road is equipped and on the way through information is transmitted to the train. Furthermore a train fitted with complete ERTMS/ETCS equipment and functionality can operate on any ETCS route without any technical restrictions. The four levels identified by standard are the following:

Level 0 = level corresponding to an ETCS vehicle running on a non-ETCS route. In this case the track side signals must be still observed and the train borne equipment has the only task of monitoring the train speed.

Level 1 = the ERTMS/ETCS system is superimposed to the traditional signalling equipment. In this case the traditional track side occupancy controlling devices (track circuits) detect the train position and are linked with the interlocking. Euro balises are usually used to transmit from track to trains fixed or variable data as the permission to cross one or more block sections (Movement Authority). With these data the on-board equipment continuously monitors and calculates the maximum speed and the braking curve. Optionally, Euro loops or radio infill units may be used.

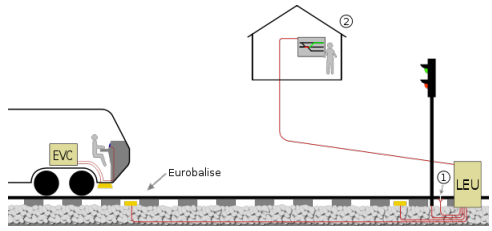


Figure 1.1: ETCS Level 1 source: Wikipedia

Level 2 = the ERTMS/ETCS system uses a GSM-R radio channel allowing a continuous exchange of data between the track side equipment (Radio Block Centre (RBC)) and the trains. Train movements are monitored continually by the RBC that generates the correct movement authorities for the different trains in the section. MA are then transmitted to the trains

together with speed information and route data. Most of the signals are displayed in the train borne cab making the line side signals no longer strictly necessary. The on-board equipment continuously monitors the transferred data and the maximum allowed speed. The trains report their position to the Radio Block Centre via the GSM-R communication channel and use the Euro-balises as passive positioning beacons or electronic milestones. Between two positioning beacons the train determines its position via sensors. The positioning beacons are used in this case as reference points for correcting distance measurement errors. For the train integrity, supervision the traditional signalling devises (track circuits) are still kept.

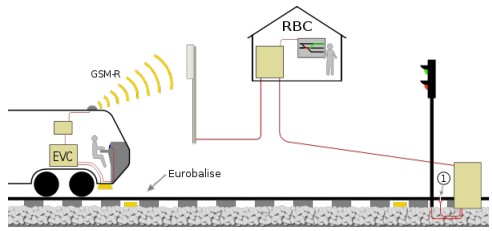


Figure 1.2: ETCS Level 2 source: Wikipedia

Level 3 = ETCS Level 3 provides an implementation of full radio-based train spacing. Therefore the line-side signals as well as the track-side occupancy checking devices are no longer required. The location of trains is determined by the train-side odometer and reported to the track-side Radio Block Centre via GSM-R radio transmission. The movement authority is then given on the information relating to the position of the train, based on the actual distance of a train from the next. Nevertheless this level is currently under development.

The continuous exchange of data between an ERTMS L2 fitted train and the track-side sub system is ensured by the GSM-R mobile communications. This mobile radio system, that is used exclusively in the railway sector, allows a constant contact between the crew and the ground (service communications and emergency management) for the exchange of data between track-side and on-board systems, by establishing a circuit-switched connection. Other communications standard have been considered by the UNISIG union. For example, the wireless and connection-oriented approach is often considered as the bottleneck of the signalling system, which considerably limits the

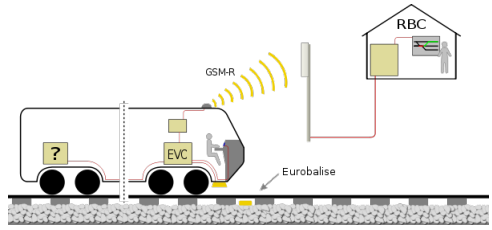


Figure 1.3: ETCS Level 3 source: Wikipedia

possible number of voice and data connections in each cell of the network at the same time and it can cause a deadlock of the system, if the number of users will rapidly increase (e.g. accidents, freight depots, lines with a high and dynamic volume of traffic). Today Spain (3800km), France (2000km), Germany (1600km) and Italy (1000km) are currently the most equipped states. In parallel to European adoption, other countries (China, Taiwan, South Korea, India, Algeria, Libya, Saudi Arabia, Mexico, New Zealand or Australia) have launched major ERTMS investment programs.

1.2 Railway Interlocking System

The main actor of railway signalling is the interlocking system sometimes indicated with acronym IXL in the specialized papers. IXL "orchestrates" the devices installed on the railway network. Moreover, its job is to apply a set of rules (further on interlocking principles) to the commands it receives from the Traffic Control (TC) and check whether or not if such request compromises the safety of the future state of the railway network. If the information it receives in form of signals do not violate the safety of the railway network then these signals are committed to the physical devices otherwise ignores it and eventually, reports to TC.

Railway network can be divided into two distinct parts: stations and line - called open line - interconnecting stations. In this work the focus is related on stations.

An area governed by an interlocking system contains railway structural elements such as railway tracks, switches, buffer stops and devices installed alongside, i.e., signals and train detectors (track circuits, axle counters and balises). Beside an interlocking area, tracks are logically divided in track

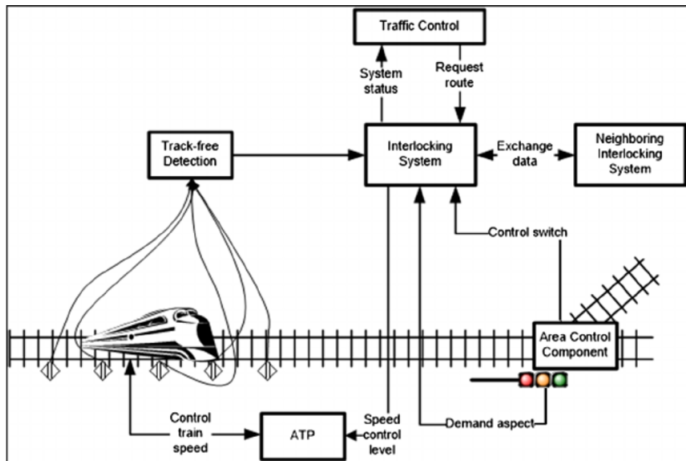


Figure 1.4: Overall architecture of railway control system, source: [39]

sections (e.g., TC1, TC2...T6 in Figure 1.5).

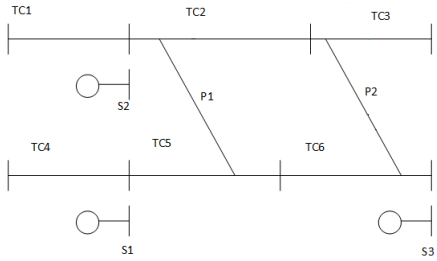


Figure 1.5: Example railway layout

Each track section is associated with at least one train detector (to identify the presence of the train on the segment), a train protection element permitting emergency brake and possibly a level crossing barrier where the railway line crosses a road. Commonly train detectors are track circuit or axle counter.

A track circuit is either cleared indicating no train on the track or occupied indicating the possible presence of a train. Typically track circuit

become larger on long straight stretches of track without any interesting topological features such as junctions or stations. Likewise track circuit become smaller around junctions and stations where control over train movement is of greater importance.

Figure 1.6: A track circuit installed along the railway section: source Wikipedia

An axle counter detects the presence and travelling direction of wheels at various points along the right of way. The right of way is broken into"blocks" with wheels (axles) being counted into and out of the block. If the same amount of axles is detected departing the block as were previously detected entering it, the block is considered vacant. Axle counter systems and track circuit systems both provide for track vacancy detection but function in significantly different ways.

Signals are the main means used to communicate information regarding the state of the track ahead of the train. Typically, they are placed either on the track side or overhanging the railway. Signals could be fixed showing information as speed limit or variables as semaphores. Visual indications known as aspects are used to convey information to the driver. A signal will have many such aspects which can be displayed, each with a particular meaning. The coloured light signal have between one - four aspects each conveying a different indication about the state of the track ahead. Below is

Figure 1.7: An axle counter installed along the railway section: source wikipedia

a description of the aspects used for a three light signal:

- Green - If this aspect is displayed it indicates that track ahead is clear for a sufficient distance and the train driver can proceed at full speed to the next signal.
- Yellow (double) - This aspect indicates the track immediately ahead in between this signal and the next is clear however the driver should proceed with caution as a train could be in the track after that.
- Red - This aspect indicates that the track ahead is not clear, the driver should stop and wait at this signal.

The two - four aspect signals are used on tracks with different speeds to convey different stopping distances. The two aspects signal for instance would be used on a low-speed track segment where stopping distances are relatively short. Whereas the four aspect signal would be used on a high-speed line where stopping distances are long and the driver needs information for a greater length of track. These signalling schemes are not fixed from country to country, for example, they may use different conventions, colours and number of lights on each signal so they will be not reported here.

A turnout sometimes called pair of points or switch (e.g., P1) are special devices connecting a track section with two or more other track sections per direction of movement. Switches have two possible positions: normal and reverse. The normal position is when the switch allows trains to travel straight over while reverse is when trains branch off.

Figure 1.8: A sempahore signal installed along the railway section: source wikipedia

Figure 1.9: A simple 2-way switch: source wikipedia

In Figure 1.10 and Table 1.1, an example is drawn.

A control table is a structured, tabular presentation of the rules governing route setting on a railway track layout. Routes are logical entities defined on track layout and consist of sequentially connected track sections that begin and end at signals. Route settings on a railway track layout derives from the principles of safe working of trains such as:

- Avoid collision: no trains occupy the same section at same time;
- No derailment: trains keep moving on rail-road, thus inhibiting the movement of mobile-elements (i.e., switches) during the train passage. From the configuration point of view, this means that all switches belonging to a route must be identified.

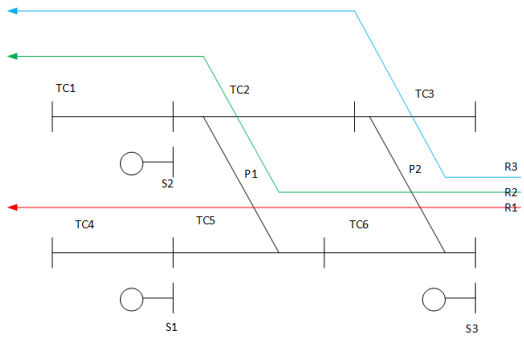


Figure 1.10: Example railway layout with three routes (respectively coloured red, green and cyan)

An entry signal shall be clear to let the train enter the route. Columns of the table indicate the:

- point settings required by the route;
- conditions for locking and releasing other routes;
- tracks required clear in order to clear the entry signal;
- route replacement rules;
- And approach locking acquisition and release.

Some signalling conditions are presented in other tables, subsidiary to the main route table. There is a point control table that indicates the conditions needed to move a point [98]. Since the interlocking connects points and routes, the two tables are, to a degree, converses of one another. Other tables cover approach locking, aspect sequencing, level crossings and dual gauge requirements.

The format and content of tables is not standardized, and may vary even within the same railway administration. Diverse safe working practices and signaling technologies drive diverse table formats [98]. Control table is also used as a test specification for the interlocking i.e. this document also acts as an agreement between the railway administrators and the railway companies [98].

A control table represents an intermediate level of design between a track plan and a wiring diagram. Each row in the table represents the requirement

Route	S IN	S OUT	Aspect	S Ahead	T	PN	PR
1	S2	S1	Y/G	R/Y	TC4, TC5, TC6	P1, P2	-
2	S2	S3	Y/G	R/Y	TC6, TC2, TC1	P2	P1
3	S2	S3	Y/G	R/Y	TC3, TC2, TC1	P1	P2

(R) Route
(S IN) Signal Input
(S OUT) Signal Output
(A) Signal Aspect
(S Ahead) Signal ahead
(T) Tracks
(PN) Point Normal
(PR) Point Reverse

Table 1.1: Example control table refereed to 1.10

how to set and release each route. For example, in table 1.1 route number 2 comprises the track circuits TC6, TC2, TC1 and requires that the points P2 are in normal position while P1 in reverse.

Although there is significant difference in the way that interlocking systems are implemented, all currently used systems provide the same functions. Moreover, when observed as black boxes, the behaviour of most interlocking types will be the same in most situations. Indeed there are different control methods but four are considered basic which are widely accepted and used among railway companies:

- Route locking: 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 (showing yellow or green).

- Approach locking: After a route is set, the point is locked and the entry signal is cleared, if the track circuit in front of (approaching) the entry signal is occupied, then the signal man cannot cancel the route and the entry signal by the normal procedure. Approach locking prevents the train driver from the sudden change of signal aspect from green or yellow to red.

- Route released: After the passage of the train, the reserved route is released automatically.

- Flank protection: The equipment within the surrounding area of the reserved route that may cause an accident shall be protected even if no train is expected to pass such a signal or such points. For example points should be in such positions that they do not give immediate access to the route.

An example of interlocking principles is illustrated here after. When the railway operator requests the IXL to set and lock a route, the IXL:

1. Checks the route signal (signal that indicates whether a train could run from a location to another location within a station) enters the route-requesting state after receiving the route-requesting information from a train;

2. Checks whether there are routes with uncompleted settings that may be in collision with the requested route, if not, go to the next step;

3. Checks the status of the tracks that the requested route will use, if unoccupied, then reserve those tracks;

4. Checks whether the points are in the required position, if not, switch to the requested position;

5. Lock the conflict routes to guarantee the safety of the requested route;

6. Checks whether the tracks and points related to the routes are usable; if usable, turns the route signal green, which means the train can enter the route; For the sake of avoiding subsequent trains' entering which may lead to tail collisions, block the signals immediately as soon as the train enters the route.

The interlocking system executes the above mentioned interlocking logic that satisfies the interlocking principles and the control table for the particular station layout. As and when a train moves along a route, the IXL system releases the locked devices behind the train and lights the aspect of the signal at the entrance of the route forbidding the trains to enter that route. Another example of IXL principle is the rule that governs the control of points.

As it was said above a control table is an off-line document that exists with the track layout schema relative to a station that is compiled and update by the railway authorities' engineers. The pervasive introduction of the computers reflects also on this subject.

It is clear that the process of generating a control tables has some routine elements that are amenable to automation. It does not appear to be feasible to generate a complete table automatically. Some signaling rules are imprecisely defined, some are difficult to capture in the control table format, and at some locations there are special rules that deviate from the general principles [98].

It is possible to identify 4 main processing units:

- Track Layout Editor;
- Control Table Generator;
- Control Table Editor;
- Control Table Verifier.

The Track Layout Editor is used to produce a graphical representation of a track layout. The user draws the track on the screen and decorates it with the elements needed for a control table (track segments, points, signals etc.). The editor undertakes consistency checks on the drawing (e.g., that precisely three track segments join at a point) and writes out the elements and their connectivity in a formal notation [98].

The Control Table Generator is a tool that generates control table entries for a layout, given its XML description. It generates all entries of the table that can be inferred from the layout. However, in general it is not possible to compute the entire table automatically. Limitations include situations where different options are possible, or where extra commentary or notation is needed to fully specify the meaning of a condition. The output of the tool is a structured description of signalling controls for the layout. It is

expressed in XML, in a form that mirrors the railway authorities control table structure [98]. The Control Table Editor allows the user to add, remove or modify entries in a Control Table. Control Tables are presented in a style similar to a spreadsheet. Edited control tables are stored in the same interchange format produced by the Generator. User editing is essential, for several reasons. First, it is still sometimes desirable to produce a table by hand from scratch. Second, as indicated above, the Control Table Generator will not always produce a complete table. Third, specific local signalling rules apply at some locations. These local rules may be either more or less restrictive than the general principles. From the Control Table Editor, the user can perform basic consistency checks on the Control Table. For example there might be a check that each point control in the route table has a corresponding control in the points table [98].

The Control Table Verifier is a tool that checks a control table against a set of Signalling Principles. To achieve this it translates a Control Table into a model that defines the behaviour of the signalling objects (e.g. points and signals) for the layout, based on generic Control Table Semantics, which define the meaning of the Control Tables [98].

1.2.1 Relay Interlocking and Computer Based Interlocking

The most common technology used in the past was based on relays, namely Route Relays Interlocking (RRI). These relay based circuits implement all types of interlocking logic, that will be discussed further, and take inputs from above cited equipment (i.e. signals, points and track Circuits) in the form of relays. Despite the fact that relays are phasing out, they are still in use in the small stations and nevertheless they had been of great importance because they were used as start point in developing the newest systems as it will be clarified in this work. The command to set, monitor and clear the route for the train is taken in the form of buttons by the station master's console (IECC in UK and ACEI in Italy are some example) or more recently by workstation equipped with control software. The output of the interlocking logic is also a relay, which in turn drives the signals and turnouts associated with points. The relays circuits are build using the station control table (sometimes called Control Laws or Conditional Tables) as input which is generally paper document and the interlocking principles as the Business Logic (further on Interlocking Logic). The control table rules the possible

movements of the train inside a station yard and its relationship with other stations.

A Computerized Interlocking System (CIS) is built using electronics replacing traditional mechanical levers and electro mechanical relays. The same control table used in RRI forms the basis here also. The relays used to form the logic circuits in RRI are replaced by software variables. In the CIS, states of track-side equipment are stored in memory. The core of that interlocking system is divided into two distinct parts: configuration and a generic interpreter of that data. Configuration data is prepared specially for each signalling scheme. The data contains both information and the application of signalling principles to that information. The source data is compiled into executable data which is interpreted at run-time. The source data is written using a special-purpose programming language which was designed by signalling engineers for signalling engineers. The above way of working meant that the first computerized systems resembled the electro-mechanical ones. This has created a considerable time lag between the new information technology (e.g. object-oriented approach) and approaches applied to systems thus constituting a major constraint for railway industries. This is also a recurrent aspect in human life, that implies low adaptation to the new technologies (the first cars, for example, resembled buggies without horses).

The language is constructed from expressions and statements. Primitive constructs of the language consist of tests and commands which are used to read and write to the interlocking memory. The field inputs are collected using digital input cards and outputs are given using digital output cards. The processing is done by a processor where the virtual relays (software variables) are evaluated using the interlocking equations, which are now in digitized form either as algorithms, Boolean equations or state charts in the processor memory. These algorithms now being executed by the processing unit take appropriate action. The advancements from tried and trusted electro-mechanical solutions - they implement the proven interlocking rules and also since the relays used in RRI are inherently fail-safe (due the fact that they drop to safe state due to gravity even when power supply is not available or in any kind of malfunction) - to modern functional equivalents of today was principally due to the next facts:

- improved availability because redundant architectures as it will be presented further on provide inherent redundancy through switch-over to duplicate systems -both processor and input/output modules;

- enhanced maintainability as the provision of extensive logging and diagnostics facilities comes as standard across most contemporary products and the most modern solutions provide predictive diagnostics to target behaviour trends, eliminating potential failures;
- reduced equipment footprint as for example, in a typical four tracks station the number of relays used to implement this type of logic would be in the order of 1000 relays and wiring is so complex that the time taken to install and commission a RRI is very long. The testing of the system requires the total station to be setup and testing done during normal train operation.
- increased communications capabilities (related to network communications) to allow simpler and higher performance of relational data streaming around the signaling and train control system;
- Power loading improvements as by design, the scale of relay interlocking is significant - up to tens of thousands of relays, individually, the electrical load is insignificant, but when multiplied draws significant current. Besides that, poor power factor and phase imbalances can result in significant heat dissipation and overrated mains supply / UPS.

Conversely, the development of CIS has once again brought issue to take care of:

1. lack of formal methods in developing the Interlocking Logic;
2. Lack of domain knowledge in signalling and traditional RRI Systems, this creates a technological gap between the software programmers and the domain consultants that leads to errors in software, which might lead to unsafe failures of the system.

The Boolean equation method is the immediate implementation of the traditional relay interlocking principles. In this method the relay circuits are implemented as Boolean equations, so there is one to one relationship between the relay circuits and the software variables, so that a railway operator can easily validate the software entries made and this method gives him sufficient confidence. This method theoretically has very high safety performance, since the control laws once written remains constant and only the station data is changed for every yard, but the actual implementation of this method has limitations:

- the control laws are not fully tested and they are not generic;
- typically these Boolean equation are in huge numbers and very difficult to verify these equations.

Most of computer based interlocking systems use (in their implementation and/or formal specification) the above form of centralized database, cited as control tables. In the 2000's, it was introduced another approach to interlocking rules mainly well discussed in [28] and [54]. The paper has addressed the problem from a "geographic" and "distributed", point of view. Fantechi et al. call the above table-based approach "functional" because in that approach the rules are generated by adopting a design methodology focused on functions, such as the switch points checking function, the routes setting function or the routes verification. In the Geographical approach the input to the interlocking systems is given as the position of the signals, points, tracks Circuits and Slots. The Interlocking is implemented based on the generic rules such as no part of the track are shared by the two routes at a time, Conflicting routes should not be set at a time etc. This type of implementation requires a great knowledge of the Yard Elements and the interconnection between them. In this method the software does not have one to one relationship to the relay circuits used for RRI and is very difficult to validate, so this method has failed to create the necessary confidence in the railway operators. They also noticed that placement of the yard equipment is ignored in this design methodology, it does not exist a direct correlation between the geographical position on the yard of a device and the function that controls it implemented in the IXL. They also pointed that it is a hard task to identify the parts of the system stimulated by external events. They also introduced a different approach that can consider distributing the knowledge of the interlocking rules to objects modelling the geographical placement of physical elements. This approach has broadened the field of design of IXL including now the object oriented approach.

1.3 Verification and validation of Railway Interlocking system: Literature review

Railway interlocking system are Safety-critical systems because failure results in loss of life, injury or damage to the environment. The conventional verification methodology based on factory and on-site tests discloses most

of the errors but it cannot claim that it discloses all the errors. The requirements of an interlocking system can be divided into two categories: generic and specific requirements. An example of a generic requirement is a condition that must hold whenever a route is cleared:

"A route must be cleared only if every switch in the route is proved in the position required by the route"

A specific requirement applies to a single interlocking system. An example of such a requirement is the configuration of the actual routes in the system:

"Route XYZ requires that switches 1X and 1Y are normal and that switches 2X and 2Y are reverse"

A clear and complete generic requirements specification gives a precise description of the functionality of a range of systems and can be used when validating i.e. that the developed system meets all requirements. However generic requirements specifications are typically written in natural language.

Natural language may seem easy to understand at first glance, but experience shows that it tends to be interpreted differently by different people. Indeed, in reality the quality of generic requirements specifications is often unsatisfactory. Although requirements often are captured in documents or perhaps even in a requirements management tool, they are typically not complete, and the successful development relies on implicit requirements, existing only in the minds of the engineers.

As a consequence, it is hard to guarantee that systems are designed in a consistent way. An efficient way to resolve possible ambiguities in requirements is to instead express them in a formal language. A formal language is a mathematical language, in which every statement has exactly one, well defined, meaning, much as in any well-defined programming language.

IXL software cycle-of-life is ruled by national guidelines. In Europe, CENELEC EN 50128 is the European standard specifying procedures and technical requirements for the development of software for railway control and protection systems.

The CENELEC EN 50128 Standard, "Railway Applications: Software for Railway Control and Protection Systems" is part of a group of related Standards: the EN 50129 and the EN 50126 [58] [56] [57].

The latter ensures that the embedded software is suitable for use in safety-critical settings, while the EN 50129 provides guidelines for the electronics systems used for signaling.

The EN 50126 and the EN 50129 are focused on the safety functions allocated to software, whereas the EN 50128 specifies the methods which need to be used in order to develop software which meets the demands for safety. The key element of this latter norm is the concept of software safety integrity levels (SILs).

SIL defines the robustness degree in terms of protection level that the system must have against the failures both random type that systematic type. There are four safety integrity levels (from SIL1 to SIL4): SIL4 is associated with the highest level of integrity, while SIL1 is associated with the lowest. The SIL0 value is used for indicate that there are NO safety requirements.

For the two highest safety integrity levels, EN 50128 highly recommends to use of formal methods for software requirements specification, software design, verification, and software validation.

Formal methods are mathematically based techniques and tools for the synthesis (i.e. development) and analysis of software systems. Formal methods can be applied at various points through the software development cycle i.e. software specification and software verification.

Formal methods can also be used in reverse engineering to model and analyse existing systems [67]. Initially formal methods were promoted aggressively by idealistic academics. Formal methods were seen as the solution to the "software crisis" and implied the uncompromising use of mathematics and rigor for the whole development life cycle where entire software systems were developed from scratch.

Hence, the development process should start with an abstract specification of the system and proceed by formal refinements and proofs of correctness towards a final implementation [25].

From an industrial point of view this is not realistic because systems are rarely developed from scratch, only parts of the systems would benefit from a formal model and the general skill level in industry is not adequate to cope with the techniques for fully formal development.

Many companies were reluctant to throw away current practice but they began to allocate massive investments required by such a revolutionary approach [25] .

With this approach formal methods are used more as a defect detection technique in the early stages of the software development life cycle. Hence the abstraction inherently associated with formal modeling is a powerful means

of reducing complexity and improving a development team's understanding of the requirements.

Conversely formal notations provide a structured way of expressing requirements which makes inspections more effective and reliable and allow different sorts of checking to be performed by automated tools [25].

Before formal verification of interlocking system software, the system must first be formalized. This requires a computational model of the interlocking. The model includes a language (syntax and semantics) used to program the interlocking and relevant hardware. The hardware that needs to be formalized depends on the interlocking design. For example, timers are part of the hardware, but not necessarily directly supported by the language.

Over the past 30 years, there have been many attempts to formally model various aspects of railways. Some, attempt to capture as much information as possible while creating the models, these models are typically written using a specification language such as VDM, RSL, B/Event-B or Casl.

A well-known example of this algebraic specification approach is the Paris metro, Metro EST Ouest Rapide (METEOR) [31] [76]. METEOR is a driverless metro line that was partly built by Matra Transport International. Matra Transport decided to use the B-Method during development. B is a formal language which is the successor to Z. It is a complex language to use partly because the specification and implementation are deeply intertwined which requires successive refinements to obtain an implementation. Perhaps Matra Transport decided to use the B-Method as it was developed in France, so they had a wealth of local expertise. Using the B-Method Matra Transport was able to verify 100% of the safety and liveliness requirements of the ATP/ATO systems. It was claimed that no bugs were found during the validation, in house testing, on-site testing and since METEOR went live. This is clearly an exceptionally meritorious result for such a complicated, critical system [72].

Other work relating to the specification of railways acutely focus on the topology and operational aspects. For example, the approach taken by uses graph theory by Hansen [6]. The basic idea is that track segments are vertex, and the edges are connections between the segments. The edges are directed hence they must be doubly linked. The reason they are directed is that it allows the edges to be annotated in only one direction. For example, signals are added to the model by annotating edges, this allows the model to represent that the signal is only visible in one direction (which is a requirement

in the railway domain).

Hansen specified Danish interlocking systems using the Vienna Development Method (VDM) specification language. This specification was simulated so that domain experts could validate its correctness. The first attempt omitted manual overrides that are used for shunting and controlling sets of points. The second attempt was augmented to provide these manual overrides, then verified against the safety requirement that the interlocking could not allow trains to collide or derail. Although safety conditions that are more precise were not verified, this high-level requirement is the ultimate goal of the interlocking with respect to safety (not liveliness) properties. It is clear from Hansen's work that using a graph as an underlying data-type to represent the topology is flexible and extensible. There are a number of different operations that can be applied, for example, the vertex relating to routes can be colored to determine incompatible combinations, or the graph can be traversed to determine whether the topology is well-formed.

Another approach is to use predicates. They express a richer view of information than the graph based approach. Predicate logic has been successfully applied in several projects to model railways [52]. It is straightforward to define n-ary relations between objects: a Prolog-like syntax is suited well to this. For example, this is the approach taken by Eriksson [52] when modelling the Swedish rail topology. Using objects and relations between these objects, the physical world and abstract concepts (such as routes) can be formally modelled. Provided the model adheres to a suitable nomenclature the model will be readable by humans. Having a formal model of the topology allows for information to be automatically deduced, e.g. pairs of incompatible routes.

Currently there are two main approaches to the problem of verifying the correctness of safety critical systems using formal methods i.e. model checking and theorem proving. The model checking is the approach most widely used by industry. The model checking problem is as follows: given a model and formula expressing some property we would like the model to have, final under what conditions does the model satisfy the formula [75].

The two main techniques typically used in model checking are SAT-based model checking and binary decision diagrams. Using model checking has the advantage over deductive methods that verification is automatic. However not every system or property can be verified by this method due to the limits of what is decidable /computable. One further advantage is that a model

checking algorithm will typically provide a counter example trace. These traces show the exact sequence of states which brought about an error. This property is useful because it can be used by engineers to find faults in the system.

Early model checking procedure verified models formalized in CTL. Computation tree logic introduced by Clarke et al [45] is a branching-time logic, meaning that its model of time is a tree-like structure in which the future is not determined; there are different paths in the future, any one of which might be an actual path that is realized. Nowadays, time logic is used for formal specification, validation and verification for real time systems. A notable review of those logic languages that discern by expressiveness, order and metric of time is [34]. One the most well known problems in the field of model checking is state space explosion. The number of states in a model of a concurrent system increases exponentially in relation to the increase in number of components in the system. The main focus of the further research has been to reduce the number of states. This has been achieved through the use of state based reduction, path based reduction and compositional reasoning. A simple form of state based reduction would be to remove a state from the search space if it is bisimilar to another state already encountered. Other methods include program slicing which simplifies the model with respect to the safety property being verified. The obtained model only contains information about the system that is relevant to the safety property. Progress has also been made in reducing the amount of physical storage needed for a model using caching and compression. While these techniques cannot prove an entire system correct they are useful for finding errors.

The formal verification of railway interlocking systems software has been studied a number of times over the past 20 years. Most research into verification of interlocking systems has focused on using exhaustive techniques: SAT-solving, symbolic model checking and explicit state model checking.

Typically, case studies focus on a specific interlocking installation and try to prove that the interlocking installation will never allow train movements that might lead to collisions or derailments.

An approach used in multiple case studies is to translate the interlocking logic to a model (for model checking) or a Boolean equation (for SAT-solving).

Properties are defined as formulae, representing requirements over the inputs and outputs of the system. These properties can mostly be derived

directly from the track layout.

Model checkers or SAT-solvers are then used to prove that the desired properties hold for the models or equations. The first verification attempts focused on determining whether an interlocking system satisfied concrete safety properties, i.e. formulae represented in terms of the input and output variables of the system.

During the mid-nineties, a notable body of work relating to the verification of the interlocking systems at Hoorn-Kersenboogerd and Heerhugowaard (Netherlands) was undertaken by Fokkink [55]. Years later, the same stations/problem sets of Hoorn-Kersenboogerd and Heerhugowaard were revisited by Eisner in [51], where above cited CTL model-checking was successfully applied in-place of SAT solving. It is noted that the SAT based techniques were successfully applied in Sweden by Eriksson [52], where a serious design flaw in the interlocking system was identified.

Orthogonally during the mid-nineties an effort was made to apply model-checking to a more complicated class of interlocking systems that are programmed using the Geographic Data language [80]. This notable piece of work applies the Calculus of Communicating Systems (CCS) and mu-calculus to verify safety properties of Westinghouse's Solid State Interlocking (SSI). Morley defined how the interlocking and its interpreter can be modelled using CCS and also defined a translation from the SSI's command set into CCS, thus creating a complete model of the interlocking.

Then using mu-calculus and the Concurrency Workbench, he verified safety properties.

The project also verified the communications between interlocking systems. An example of this interlocking communication occurs when a train route is spanned through. Mu-calculus can also be used for liveness properties. Multiple interlocking systems and the interlocking systems must communicate information regarding the whereabouts of the train and route status. The transition from verifying a single interlocking system, to verifying the communications between multiple interlocking systems was greatly simplified through the use of CCS. The verifications described thus far do not make safety verification tractable. They only provide a mechanism to perform verification of safety and/or liveness properties for interlocking systems.

In [27] was applied the state charts formalism to perform safety verification on distributed interlocking systems. Notably, in that work the high-level formalizations of distributed interlocking systems are introduced.

In [55] demonstrate the use of domain specific languages in the railway domain. This includes defining a graphical language that the topology can be represented within, then automatically deriving from one of these diagrams the required constraints for the network to be safe. This information is then automatically translated into a control table, i.e. a high-level specification of an interlocking system. Typically, a specification of the railway is made using a formal specification language such as RSL, and then some refinement method is applied to produce a working system that fulfils the specification. Assuming the specification has been validated to be correct, and then the control system will meet these conditions, as well. This is the procedure followed by the previously mentioned METEOR project.

Verification can then proceed by deciding which states of the track-side functional modules were in conflict, e.g. two opposing signals should not both show the proceed aspect. All possible transitions into these bad states are computed from the state machines, these transitions are then used to build a "safety monitor" state machine.

The formal development of the distributed railway control systems was performed in a more traditional manner. The specifications were made executable, then using these high-level specifications of the distributed railway domain, safety requirements such as "no collisions" and "no derailments" were verified. Subsequent refinements were made preserving these properties until a working system was produced. Verification is carried out on multiple levels, one level of interest are control tables. It is suggested that model checking could be used for this verification in. [55] also suggests that the generated code can be automatically verified against the control tables.

With respect to state machines, the most popular formalisms, as represented in the research literature, are Unified Modeling Language UML statechart diagrams (as specified in UML 2.0), because the UML is a widely accepted modelling standard in industry. Without extensions, however, UML does not allow modelling and evaluating properties like timeliness, throughput, or fault tolerance. The modelling facilities provided by the conventional state machine modelling technique are used frequently to model real time systems.

Interlocking systems are based upon events happening to which the system responds, with a transition to another state. This directly maps onto the state machine model for simpler systems. State Machine Modelling is concerned with modelling the system describing it as a collection of discrete

sates. State machines model these states in a language independent manner. The state model of a system is always in one of its set of possible states, the model transitions from one state to another system state when stimuli are received.

There are limitations to the use of conventional state machine modelling. One of the limitations inherent in conventional state machine modelling techniques is that the complexity of the state diagram increases dramatically as the number of possible states increases. The state machine model then becomes unwieldy to use, this detracts from the intended purposes of state machine models, to allow the system to be modelled in a more comprehensible manner. Another of the limitations of modelling a computer system as a conventional state machine is the lack of support for concurrent constructs. Traditional state machine modelling is based on sequential transitions from one state to the next. Concurrent systems cannot be modelled in this manner as various aspects of the system may be in different states. It is seen that the limitations inherent in state machine models include the complexity which occurs as the number of states increases, and also the modelling of concurrent systems.

Statecharts overcome the limitations of state machines by providing a construct known as the and-state. In the Mealy-Moore state machine model, states were an or-state; the machines was either in one state or another, never in two states concurrently. The and-state overcomes this limitation of or-states by allowing the state chart to have substates of a higher level state active at the same time. The state machine model could be decomposed into lower states, however these lower states were still sequential not contemporaneous, in the state chart modelling system proposed by Harel these substates may be concurrent. Each of these substates models an aspect of the object of which they are a part.

These substates are also allowed to communicate with each other in well-defined manners. These communication and synchronization methods have been enumerated by Douglass [50]

1. All orthogonal regions of the chart accept events sent to the object

2. One region may create an event as a result of a transition that is consumed by another orthogonal region.

3. A guard may be used to test if another region is in a certain state before allowing a transition to occur to the guarded state.

The third enumerated point introduces the concept of a guard. This is a construct introduced to Statecharts which adds additional features not available in a state machine model, conditional event responses.

To overcome the second limitation of conventional state machine modelling, that of a quickly escalating level of complexity as the number of states increased, orthogonal regions are all used. These regions allow the modeler to detail highly complex specification logic in a condensed manner. Harel [66] stated "Modelers can use orthogonal states to decompose large state spaces naturally into independent (or almost independent) parts."

This flexibility enable statecharts to be far less complex: as the number of states increases the modeler can use some of the more powerful facilities provided by statecharts, such as orthogonal regions and guards, to succinctly specify the required logic in a more comprehensible manner.

A brief overview of statecharts has now been provided, showing how the statechart method of system modelling overcomes the difficulties that may be experienced when attempting to model real-time systems using state machines.

CSP (Communicating Sequential Process) is another commonly used method of specifying the behavior of modelled systems. It is also used to model the flow of control from one "state" to another, and also supports concurrency.

A fundamental aspect of CSP as defined by Hoare [70] involves the synchronization of concurrent processes on their inputs and outputs. The synchronization only occurs when two processes wishing to synchronize to send a message, are both in a sending and receiving state respectively.

Statecharts also make provision for the synchronization of states or processes with the functionality provided by guards. These guards are placed at the transition from one state to another. The transition may only occur when the transition event occurs and the condition on the guard is met, a conditional event response. This condition may be that another orthogonal region needs to be in a particular state. Thus these two separate states may be synchronized. Both techniques are based upon mathematical foundations. This enables a degree of formalism to be inherent in designs produced according to these two methods, formal reasoning is therefore possible with both of these practices.

CSP and statecharts do provide similar functionality in many areas; this is to be expected due to their predominant use in developing real time sys-

tems. There are some differences within each of their respective paradigms. Contrasting CSP and statecharts on a paradigmatic approach statecharts are concerned primarily with clearly and succinctly express complex logic relationships between states.

Hilderink [69] showed that CSP is primarily concerned with the specification of communicating concurrent processes, and describing complex communication patterns. CSP is not suitable for describing the internals behavior of processes, whereas statecharts can address this.

The method by which CSP and statecharts describe the specification is also different. Statecharts provide a graphical representation of the system. CSP provides a character based representation. These two representations each have different advantages and disadvantages, a statechart may be more comprehensible to many people, a CSP description may be easier to include as a comment in source code thereby enabling the programmer easy access to the specification to which he is writing in an easy and unambiguous manner.

Statecharts are also unaware of the concept of time in the same respect as timed CSP. Although statecharts may recognize that transitions take some time (unlike state machines), they do not include the notion of states taking a measured amount of time in the same respect as timed CSP does. When developing real time systems timed CSP provides a better set of tools for dealing with "real time" than statecharts do. Statecharts were introduced as a development over modelling techniques such as traditional state machine modelling. The weaknesses inherent in state machine modelling such as overly complex diagrams for larger systems, and the lack of concurrency support were addressed by the statechart method. Statecharts overcame these weaknesses by introducing concepts such as orthogonal regions, and-states and also guards. These concepts allow for the specification of far more complex real time systems than traditional state machine modelling.

1.4 Conclusion

The Developing of new IXL systems is ruled by normative i.e. EN50128 for European countries winch impose the use of formal methods for specification and verification because IXL are critical systems, i.e. their failure can cause loss of human life. In academic, formal methods has been successfully applied on railways and especially on existent railway signalling systems as has been presented in literature review. Conversely, applying and using formal method

needs specific expertise while model checking techniques when used on big system such as large stations suffer the state explosion problem as describe in [46].

In chapter 3 will be discussed a proposal to tackle this issue grounded on theory described in chapter2.

Chapter 2

Knowledge Based Systems and the Semantic Web

This chapter gives the theoretical foundation of the proposed solution model. In particular, in section 2.1 knowledge base systems are introduced, in section 2.2 semantic web are presented and finally in section 2.3 some bare-bone ontologies and related tools are shown.

2.1 Knowledge base systems

Knowledge-base systems are computer programs designed to emulate the decision-making capabilities and knowledge of human experts in a specific domain to address a well-defined problem [29].

Knowledge-base systems can be considered as a special type of database that holds information representing the expertise of a particular domain [79]. As depicted in figure 2.1 a Knowledge-base system consists of a knowledge-base, decision rules and an inference engine. The knowledge-base includes knowledge i.e. facts, and data about a specific application. Decision rules are heuristics and quantitative criteria developed by human experts based on through knowledge of a system, to reach a conclusion such as the status of a batch run, given the facts such as process data [29]. The inference engine processes the rules based on knowledge about the system, data, and conclusions reached by other rules (inferences) to deduce a conclusion.

Until the last decade, the main approach to build Knowledge-base system relies on dialects of prolog or lisp languages known as Rule based systems. Rule based systems are knowledge based systems where each rules can be expressed by *if-Then* statements. The if-part is called the antecedent. It consists of one or more of condition elements. The representation of the conditions may be categorized for simple problems, integer/real intervals or combination of these for more complex problems. The then-part is called consequent or action and consists of number of actions.

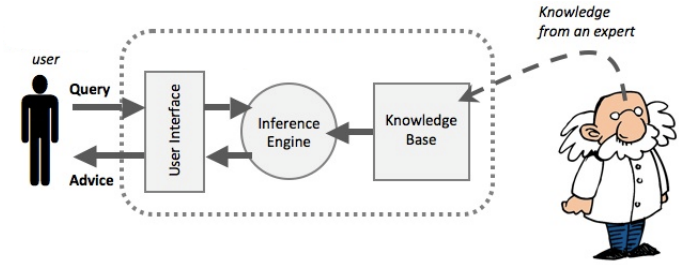


Figure 2.1: Simple diagram of knowledge-base system (source internet)

In recent years, with the beginning of the semantic web - which will be discussed here after - a new interest has been aroused in the knowledge description formalisms, i.e ontologies. In computer science, an ontology is a method to formally represent a set of concepts. Ontology is concerned, in fact, of the study of being, or of what it is and its basic categories as proposed Gruber in [64].

An ontology can be used to build knowledge bases i.e a knowledge base is an ontology with a set of instances [82]. In computer science, ontologies are most used in the field of Artificial Intelligence, for the classification of the data; Gruber defines an ontology as a "specification of a conceptualization" [64]. He states that a formal representation of a set of knowledge is a conceptualization, i.e. a set of objects, concepts and relationships between them that exist in a particular area of interest. A conceptualization is, therefore, a simplified and abstract representation of the particular field of knowledge to be represented, for any purpose.

An ontology is thus an attempt to formulate a exhaustive and rigorous conceptual scheme of a given domain, and this pattern can take several forms, in fact, there are different types of ontologies, and below, a classification based on the language expressivity and formality, is explained [89]:

- Information Ontology: it is composed of diagrams and sketches used to clarify and organize the idea of collaborators during the development of a project. This type of ontologies are only used by humans and among their characteristics, there are the ease of modification, scalability, conciseness, schematization. Information ontologies focus on concepts, instances and their relationship, and they are mainly described by means of visual languages.

- Linguistic Ontology: dictionaries, vocabularies, dictionaries, taxonomies, folksonomies, thesaurii and lexical database, are some examples of this type of ontologies, that mainly focus on terms and their relationships. Their aim is to present and define the used vocabulary, which is the results of a terminology agreement between the interested users' community. SKOS (Simple Knowledge Organization System) [5] is the preferred language to describe Linguistic ontology.

- Software Ontology: this type of ontologies provide conceptual schemata whose main focus is normally on data storage and data manipulation, and they are used during software development activities, with the aim to guaranteeing data consistency. Software ontologies are normally defined with conceptual modelling languages used in software and database engineering like Entity-Relationship Model Language or Unified Model Language (UML)

- Formal Ontology: they require a clear semantics for the used language in defining concept, and strict rules about how to define concepts and relationships. All this can be obtained by using first order logic [95] or Description Logic [26] where the meaning of each concept is guaranteed by formal semantics. An example of this type of ontologies, is a Knowledge Base (KB), that is a formal system to capture the meaning of an adopted vocabulary via logical definitions. The main purpose of formal ontology is reasoning.

At the basis of each ontological model are the concepts of *class* and *property*. A *class* is a category, a set or a collection of individuals. Different modeling languages provide different constructs to define classes. For example, it is possible to assert that a class is a specialization of another class (Dog and Cat are Animal subclasses). Classes can also be defined by means of constructive forms such as union, intersection, difference, or complement

of other classes.*Properties* are attributable to the elements populating ontological models. By means of these it is possible to describe more accurately the structure of the elements of a class. Elements of the Person class have a name (*have a name* is a property) and have parents (property that links different *individuals*).

It is also possible to define characteristics related to the property itself. For example, you can specify that *have mother* is a specialty of *having a parent*; That *having a parent* admits the inverse property *having a child* or that property *having a friend* is reflective. Classes and properties are elements common to all ontological modeling languages and allow the description of the concepts contained in the domain, or its *intensional* component. Another element of the model are individuals; they are instances of a class, for example the individual "Paul" is a Person. Parallel to individuals, there are corresponding concrete realization of properties, for example Paolo *has a surname* that is "Rossi". Instances and their attributes constitute the *extensional* part of an ontological model. It is evident that there is a remarkable similarity between an ontological model and an object model, but there are also important differences. First of all, in an ontological model there are no methods: the ontologies are therefore a technology for the description of data and not for the description of algorithms. Another difference, deeper but less obvious, is the relationship between instances and classes: an individual, in an ontological model, can not only exist without belonging to a class but may also change class over time. It is possible to create an instance, assign its attributes, and based on these, deduct the membership class, which an Object Oriented programming language does not allow.

Many advantages related to the use of ontologies derive from the ability to apply to them a Reasoner, a software component capable of solving queries, validating a model, and inferring new knowledge. By providing a Reasoner with a query and an ontological model, it will be able to extract that portion of the model that meets the criteria and constraints. Reasoning can be roughly defined as a process from which we derive new information using an already existing set of data. In general reasoning can be divided in either deductive or inductive reasoning. Given the expressive power of this type of query, much higher compared to, for example, the traditional SQL language, it is possible to talk about semantic queries. Validating a model means automatically checking if the set of assertions that it contains is consistent. For example, the individual Paul could have been declared Man

in a Model and Woman in another; if Man and Woman have been declared disjoint classes, the model is inconsistent. The validation operation is of particular importance when you are operating in a distributed environment and you want to manipulate models derived from the union of sub-models from separate sources. A Reasoner can finally argue on an ontological model to infer new properties, thus inferring new knowledge. In a relational database if a user has entered a hundred records, a query can, at most, retrieve the same hundred records. In an ontological knowledge base a user who has made a hundred statements may find a much richer set of data, thanks to the Reasoner's contribution. For example, suppose the user has asserted that Paul has a son, "Mario"; The Reasoner, knowing that *having a son* is a property that he admits the inverse *having a parent*, might infer that Mario has as a parent Paul; This information has never been entered by the user, but deduced from the model, more precisely by the characteristics claimed on the model properties. Some Reasoners also admit that a set of rules, written in a precise syntax, are provided to them for further inference.

2.2 Semantic web

Semantic Web is an additional layer to the existing web which provides semantic information about its content. By utilizing this extra data an automated system can combine 'things' by their meaning and generate 'new knowledge' based on existing one. Semantic Web will enable machines to *comprehend* semantic documents and data [35]. In order to represent information on the Semantic Web and simultaneously make that information both syntactically and semantically interoperable across applications, it is necessary to use specific languages.

The W3C introduced several recommendations for the description of information on the web, and in general, information to be interpreted by machines. As depicted in figure 2.2 those languages are XML (eXtensible Markup Language), XML Schemas, RDF (Resource Definition Framework), and RDF Schemas, OWL.

An XML document consists of three parts: an XML declaration, a DTD or XML Schema, and an XML instance (XML document data). An XML declaration and schemas are not mandatory for an XML document. An XML declaration specifies the version and the encoding of XML being used. A DTD or XML Schema is a schema that constrains the structure of XML

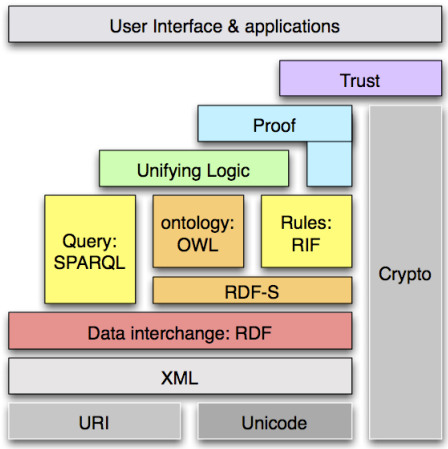


Figure 2.2: Semantic web stack, source: www.w3c.org

instances, and corresponds to an extended context-free grammar. An XML instance is a tagged document. An XML instance is a hierarchy of elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by empty element tags. Character data between start-tags and end-tags are the content of the element. XML documents have two levels of conformance: valid and well-formed. A well-formed XML document follows tagging rules prescribed in XML. An XML document is valid if it is well-formed and if the document complies with the constraints expressed in an associated schema. An XML document can be viewed as a tree, where leaf nodes correspond to data values (text) and internal nodes correspond to XML elements.

RDF is a framework to represent data about data (metadata), and a model for representing data about "things on the Web" (resources). It comprises a set of triples (O, A, V) that may be used to describe any possible relationship existing between the data - Object, Attribute and Value. RDF model just provides a domain-neutral mechanism to describe metadata, but does not define the semantics of any application domain. For example, RDF triple:

Listing 2.1: RDF triple

```
http://www.foo.bar/l.mirti,
```

```
http://xmlns.com/foaf/0.1/knows ,
http://www.foo.bar/m.zibe
```

states that the"thing" identified by URI http://www.foo.bar/l.mirti "knows" the other "thing" identified by http://www.foo.bar/m.zibe.

The"knows" predicate is also defined as an object property and it is identified by URI http://xmlns.com/foaf/0.1/knows. This property belongs to the FOAF (Friend Of A Friend) vocabulary defining aspects and characteristics of people and their relations on the web (http://www.foaf-project.org/).

For a more concise writing, a part of the URI can be considered a prefix that identifies the namespace of the thing being described. So for example "foo" could be the prefix for http://www.foo.bar/ and "foaf" for http://xmlns.com/foaf/0.1/ and thus the same triple is expressed by:

Listing 2.2: RDF triple caveut

```
foo:l.mirti foaf:knows foo:m.zibe
```

Moreover, it is also possible to state that something belongs to a class of things and also this fact can be represented by a triple. For example, the following RDF triple states that foo:l.mirti identifies something that belongs to the class of people:

Listing 2.3: RDF class assertion

```
foo:l.mirti
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
foaf:Person
```

There is also the possibility to associate simple data values (strings, numbers, dates, etc.) with a subject URI. In the following example, family name and given name of foo:l.mirti, are provided. Furthermore the familyName and givenName are called data properties:

Listing 2.4: RDF simple data value

```
foo:l.mirti foaf:familyName "Mirti"
foo:l.mirti foaf:givenName "Luca"
```

When two or more consecutive triples share the same subject URI (as in the previous example), we can write:

Listing 2.5: RDF simple data value caveut

```
foo:l.mirti
        foaf:familyName"Mirti";
        foaf:givenName"Luca"
```

A vocabulary defines the common characteristics of things belonging to classes and their relations. A vocabulary is an ontology and it is defined using the RDFS (RDF Schema) or OWL (Web Ontologoy Language). For example, the"knows" object property is defined as having as domain and range class foaf:Person. When using this information, what can be inferred is that both foo:l.mirti and foo:m.zibe belong to the class foaf:Person. Moreover, the vocabulary states that the class foaf:Person is a sub class of a more general class foaf:Agent, thus both foo:l.mirti and foo:m.zibe belong to the class foaf:Agent.

RDF Schema (RDFS) defines the vocabulary of an RDF model. It provides a mechanism to define domain-specific properties and classes of resources to which those properties can be applied, using a set of basic modeling primitives (class, subclass-of, property, subproperty-of, domain, range, type). An RDFS can be specified using RDF encoding, Figure 2(c) shows an example. However, RDFS is rather simple and it still doesn't provide exact semantics of a domain.

Listing 2.6: RDF Schema

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

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="https://www.w3schools.com/rdf/">

<rdf:Description rdf:about="https://www.w3schools.com">
  <si:title>W3Schools</si:title>
  <si:author>Jan Egil Refsnes</si:author>
</rdf:Description>

</rdf:RDF>
```

OWL (Web Ontology Language) is W3C Recommendation for representing ontologies. OWL is developed starting from description logic and

DAML+OIL. Moreover OWL is a family of three ontology languages: OWL-Lite, OWL-DL, and OWL-Full. The first two languages can be considered syntactic variants of SHIF(D) and SHOIN(D) description logics (DL), respectively, whereas the third language was designed to provide full compatibility with RDF(S).

The OWL version 2 language proposed by W3C is quite powerful; it allows the definition of disjunctive classes, union and intersection of classes, functional properties, symmetric, transitive properties, minimum and maximum cardinality of the associated elements of a property and other features. OWL 2 is still based on RDF semantics and provides datatype. OWL 2 has three profiles: OWL2 EL, OWL2 QL and OWL2 RL specifically designed and suitable for reasoning with existential quantifications, query formalization and access, reasoning and formalization of rules, respectively (http://www.w3.org/TR/owl2-profiles/).

In order to exploit the information encoded as a set of RDF triples, it can be stored in RDF stores which are optimized for the RDF data management and for the activation of reasoning processes on the collected knowledge. To this end, a specific query language was designed to find information on RDF store reticular information. Thus, the SPARQL (SPARQL Protocol and RDF Query Language, recursive definition (http://www.w3.org/TR/2004/WD-rdf-sparql-query- 20041012/)) uses an advanced matching algorithm to match portion of the RDF graph with a specified template. For example, the following query lists names of people known by a person (identified with his email) directly and indirectly through one or more people:

Listing 2.7: sample code

```
SELECT ?n WHERE {
        ?p1 foaf:mbox <mailto:pbellini@unifi.it>.
        ?p1 rdf:knows+ ?p2.
        ?p2 foaf:name ?n.
}
```

Each construct of OWL language has a formal specification in DL (Description Logic) which is also a family of knowledge representation languages extending First Order Logic (FOL). Open World Assumption (OWA) and the absence of Unique Name Assumption (UNA) characterizes OWL because enables an OWL reasoner to infer new knowledge. In an information system, OWA states that a relation not explicitly defined among elements is always

deducible. A RDF store is a framework used for storing and querying RDF data. It provides a mechanism for persistent storage and access of RDF graphs.

Listing 2.8: OWL sample code

```
<!-- http://www.disit.org/raiso#has_signal_aspect -->

    <owl:ObjectProperty rdf:about="&raiso;has_signal_aspect">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="&raiso;Railway_Sempahore_Aspect"
        <rdfs:domain rdf:resource="&raiso;Signal"/>
    </owl:ObjectProperty>



    <!-- http://www.disit.org/raiso#has_signal_begin -->

    <owl:ObjectProperty rdf:about="&raiso;has_signal_begin">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="&raiso;Home_Signal"/>
        <rdfs:domain rdf:resource="&raiso;Railway_Line_Route"/>
    </owl:ObjectProperty>



    <!-- http://www.disit.org/raiso#has_speed_changes -->

    <owl:ObjectProperty rdf:about="&raiso;has_speed_changes">
        <rdfs:range rdf:resource="&raiso;Speed_Changes"/>
        <rdfs:domain rdf:resource="&raiso;Track_Section"/>
        <rdfs:subPropertyOf rdf:resource="&raiso;has_track_charact
        <owl:inverseOf rdf:resource="&raiso;speed_changes_related_
    </owl:ObjectProperty>
```

2.3 Backbone Ontologies

It is generally accepted that building ontologies from scratch is a challenging, time-consuming and error-prone task. As an Object-Oriented language, Ontology reuse can be defined as the process in which existing knowledge is used as input to generate new ontologies. Hereafter are presented three ontologies which have been successfully used as bricks which define new ones, of which [33], [87], [59] are examples.

2.3.1 OTN

OTN (Open Transport Networks), which was defined within the framework of the REWERSE project [86], 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 [36]:

- 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).

2.3.2 SSN ontology

On the Web there are also many ontologies related to sensor networks, such as the SemanticSensorNetwork Ontology, which provides elements for the description of sensors and their observations [SSN]. This ontology has been defined using OWL 2 ontology language and it has been created starting from the review of standard and existing ontology. SSN (Sematic Sensor Network) ontology enables expressive representation of sensors, sensor observations, and knowledge of the environment and SSN is built around the Stimulus-Sensor-Observation (SSO) pattern which is an ontology design pattern describing the relationships between sensors, stimulus, and observations [47]. In SSN an observation is the result of estimating a value of a property of a

feature, using a sensing method implemented by a sensor. The SSN ontology is organized, conceptually but not physically, into ten modules; the full ontology consists of 41 concepts and 39 object properties, directly inheriting from 11 DUL (DOLCE Ultra Light) concepts and 14 DUL object properties. The ontology is able to describe sensors, the accuracy and capabilities of such sensors, observations and methods used for sensing. Also concepts for operating and survival ranges can be easily represented, as these are often part of a given specification for a sensor, along with its performance within those ranges [47].

- A sensor perspective, focused on what senses, how it senses, and what is sensed;
- An observation perspective, focused instead on observation data and related metadata;
- A system perspective, focusing on systems of sensors and deployments;
- A feature and property perspective, focusing on what senses a particular property.

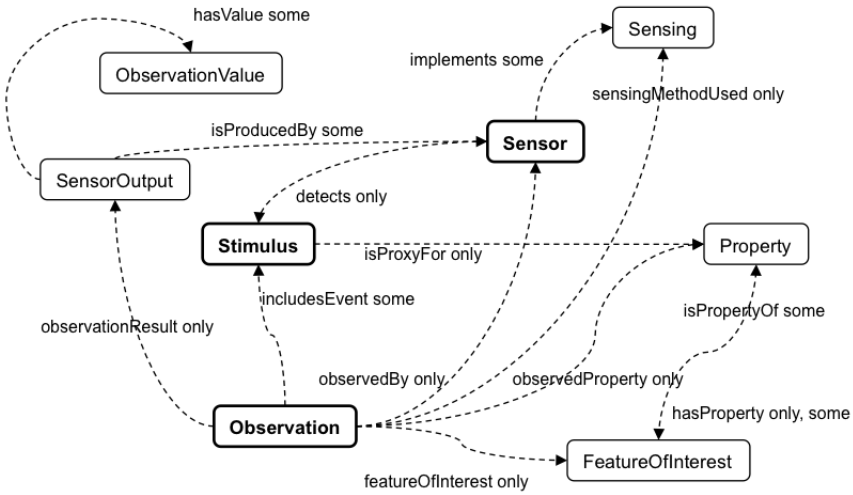


Figure 2.3: SSN, source: [47]

SSN was chosen because railway signaling devices form a large sensor network.

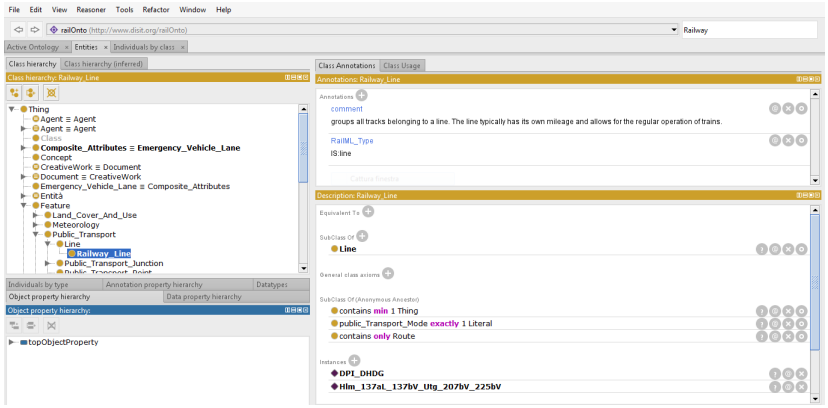
2.3.3 TIME

TIME W3C has been developed for describing the temporal content of Web pages and the temporal properties of Web services. https://www.w3.org/TR/owl-time/ Temporal information is important in most real world applications. For example, the date is always part of an online order. When you rent a car it is for specific dates. Events in the world occur at specific times and usually have a finite duration. Transactions occur in a sequence, with the current state of a system depending on the exact history of all the transactions. Knowledge of the temporal relationships between transactions, events, travel and orders is often critical. OWL-Time has been developed in response to this need, for describing the temporal properties of any resource denoted using a web identifier (URI), including web-pages and real-world things if desired. It focusses particularly on temporal ordering relationships. While these are implicit in all temporal descriptions, OWL-Time provides specific predicates to support, or to make explicit the results of, reasoning over the order or sequence of temporal entities. OWL-Time makes use of these encodings, but also provides representations in which the elements of a date and time are put into separately addressable resources, which can help with queries and reasoning applications. OWL-Time also supports other representations of temporal position and duration, including temporal coordinates (scaled position on a continuous temporal axis) and ordinal times (named positions or periods), as well as relaxing the expectation from the original version that dates must use the Gregorian calendar. However, OWL-Time has a particular focus on ordering relations ("temporal topology"), which are not supported explicitly in any of the date-time encodings.

2.3.4 Tools

In this section we introduce the tools and environment used to edit the ontology and setup the knowledge-base. They are used to develop new ontologies and modify existing ones. They let the author edit and develop ontologies concentrating on the domain's concepts and relationships, without worrying much about ontology-representation languages. The author can choose ontologies from a list, choose attributes and relations from another list, edit, add, remove, and merge ontologies. The output is usually produced in a specific high-level ontology-representation language such as OWL, RDF/RDFS, HTML, or in plain text.

Protege-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 Protege that used the frame-based formalism based on OKBC. Protege is an open-source environment that provides a framework for "plugins" either for specific applications or alternative editing tools and views.



Stardog [84] 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 on 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.

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". There are five type of constraint:

- Subsumption Constraints which guarantee 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.

Chapter 3

Modelling, verification and validation of Railway Interlocking System

This chapter presents the ontological model defining a railway infrastructure, routes and signalling principles on which corresponding validation and verification methods are applied. In particular section 3.2 gives literature review close to the approach used, the model is presented in section 3.3; the section 3.4 describes the developed ontology, while in section 3.5 results are shown. [1][2]

[1] The part of this chapter related to the railway ontology has been published as "Proceedings" in *Semantic Computing (ICSC)* P. Bellini, P. Nesi, and I. Zaza, Raiso: Railway infrastructures and signalling ontology for configuration management, verification and validation, 2016 IEEE Tenth International Conference on. IEEE, 2016, pp. 350-353. [32]

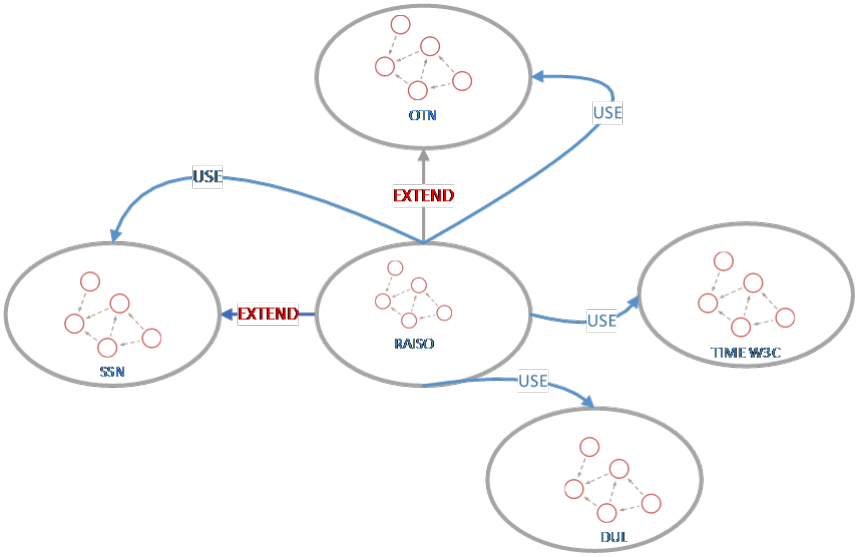
[2] *Acknowledgements:* this work was partially supported by POR CReO FESR (European Fund for Regional Development) (2007 - 2013), and confounded by the Tuscany Region, in collaboration with ECM S.p.A., and TESYSRAIL (national technological cluster on transportation 2020, technique and tool for railway transportation), supported the National Technological Cluster "Transport Italy 2020".

3.1 Introduction

Ontology development is an interactive process which involve the software engineer and the domain expert [82]. An ontology design process encompasses defining the scope of the ontology by competency questions, reuse of existent ones, enumerate important terms, define the classes and class hierarchy with their attribute, and their cardinality [65].

Raiso, which will be presented in section 3.4 is grounded on RailML infrastructure sub-schema (described in section 3.2.1) as a well-established domain knowledge and extend OTN, SSN, DUL and TIME OWL ontologies (see figure 3.1).

Figure 3.1: Raiso Modules



3.2 Literature review

According to [74], 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 ex-

change. These standardized means of exchanging information between different phases of the data preparation process are needed to fulfil the CENELEC [56] 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 [97] 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 aim of this project is a new efficient information sharing in the railway world [100] which improve decision making which also will improve performance. Moreover the consortium claims that it will be possible to identify what information has to be shared and assure that the right information can be 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 [101] 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.

An interesting academic work on ontology applied on railway domain is presented in [77]. 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. 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 work [40] 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 and 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.

In the INtegrated European Signaling System (INESS) project [88], launched in October 2008 and ended in March 2012, various data exchange tools were studied, proving that RailML has most potential. The key value of the project is a comprehensive set of interlocking specification requirements.

3.2.1 RailML

RailML is a XML based language that permits to describe railway-related data [81]. 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 sub-schema: Infrastructure, Rolling Stock and Timetable (see figure 3.2).

The infrastructure schema encompasses a complete description of physical and virtual rail elements (e.g. signals) and describes track positions according to nodes and branches modelling [38]. The rolling stock sub-schema represents the characteristics of vehicles moving on the aforementioned tracks which are divided also in ones that propel themselves and ones that cannot. Typically in such schema we 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 safety system characteristics [38]. The timetable schema shows a realizable pattern of train movement over a section of track. Re-

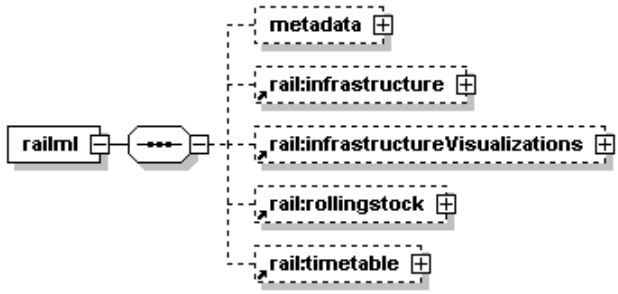


Figure 3.2: RailML sub-schema modules, source: http://www.railml.org

alizable 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. [38]

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. 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 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.

Because RailML file is as an XML file, it follows that:

- it is an hierarchical structure in which rail elements stand and ends by means of a tag
- distinguishes simple elements and complex element where the former contains sub elements while the latter contains only one attribute
- the validation occurs by checking whether it matches a fixed XSD (XML Schema Description)

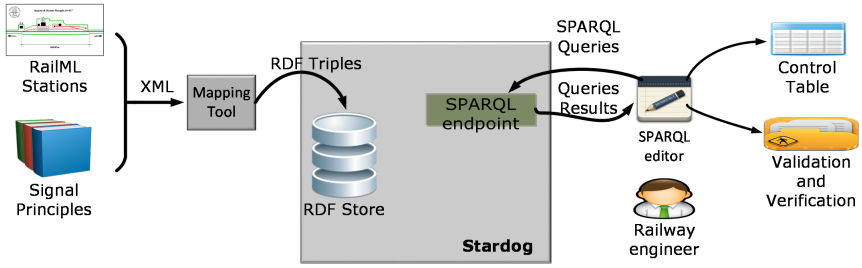


Figure 3.3: System architecture

3.3 System architecture

In this section, the reference architecture of the proposed solution is described (see Figure 3.3).

The main idea consists in creating a knowledge base model of the interlocking area under analysis. The model is created as a knowledge base grounded on the RAISO ontology model. The obtained knowledge base can be used for validating the interlocking area by verifying logical constrains.

The RAISO ontology has been modeled in OWL (Web Ontology Language) which is commonly used to explicitly represent vocabularies and their relationships in an ontology and this representation is directly usable by machine. The RAISO ontology has been represented in RDF. The knowledge-based system consists of an RDF store containing the ontology meta-model and models of railway stations. To this end, RAISO ontology has been developed and the Knowledge Base of the interlocking area into the RDF store is obtained by triples coding:

- RailML configurations converted for the RAISO ontology;
- concepts instances needed to complete the configuration model according to RAISO;
- signal principles of the specific national rules standard adopted for the interlocking area.

On the produced Knowledge Base on RDF, SPARQL queries can be applied in order to verify properties and to produce as output of the process the validated control table and reports about the interlocking system under specification.

3.4 RAISO:RAilway Infrastructures and Signaling Ontology

In this section, the structure of the developed RAISO ontology (RAilway Infrastructures and Signaling Ontology) is presented. Classes and code snippet are reported in italics.

A railway network consists of rails that can be divided into sections. These sections are of different types, such as linears, points and level crossings.Linear track sections are connected to either one or two other sections, while points sections are able to direct track in one of two direction. Linear sections must have at least one neighbour and at most two neighbours, where each neighbour of that section must be unique. A section can only have one neighbour at each end. Hence illegal configurations includes cases such as a linear section being neighbour with itself or two distinct linear sections are neighbours with each other on both ends

RAISO models railway domain at two layers: network and station. The network layer describes a railway network composed of stations (terminal or pass-through) and connections among them. This layer is realized by using the railway concepts of OTN. Indeed, *Railway_Network* class is defined as subclass of *otn* : *Railways* with object property cardinality restriction *otn* : *contains* on *Railway_Line* and *Railway_Node_Stationclasses* (see Figure 3.4).

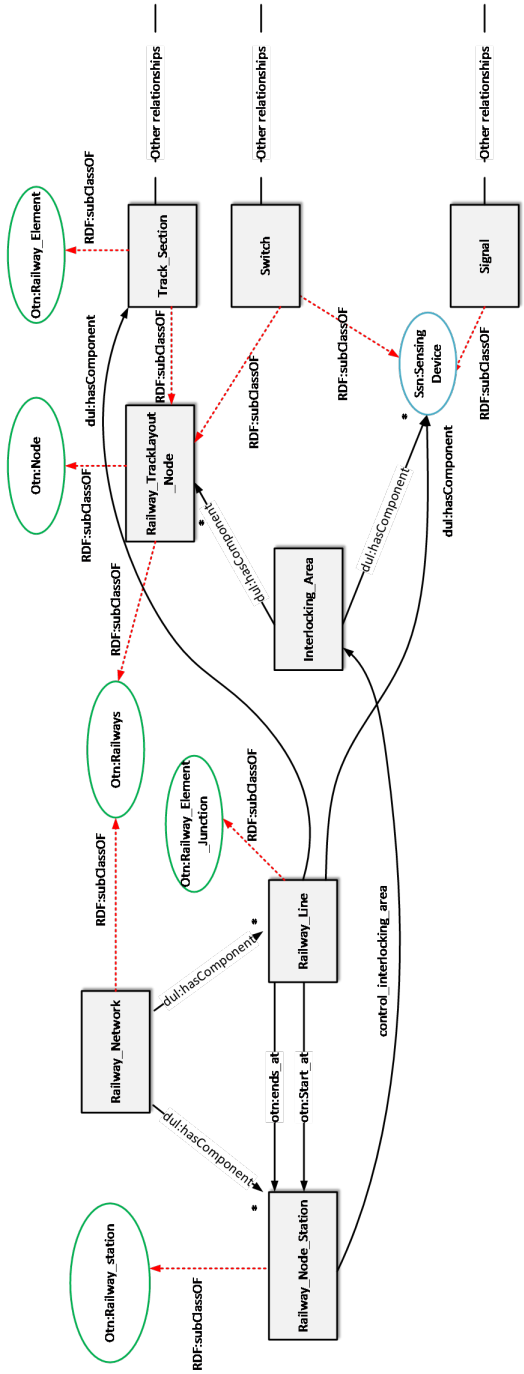


Figure 3.4: Railway Network class and subclasses

Railway_Line extends *otn* : *Railway_Element* with object properties (*otn* : *start_at*, *otn* : *ends_at*) restriction on class *Railway_Node_Station* and object property *DUL* : *hasComponenent* restriction on class *Track_Section* and *ssn* : *SensingDevice* which are detailed furthermore. Because *otn* : *Railway_Station* describes a station and it lacks of network information, we needed to extend it with *Railway_Node_Station* which is also a subclass of *otn* : *Railway_Element_Junction* and has restriction on *Interlocking_Area* class (see Figure 3.4). High-speed lines are conceptualized by *Railway_Line_HSclass*, which is a subclass of *Railway_Line* and has restriction on the speed of contained *Track_Section* typed individuals.

Station layer describes relationship between elements of a railway station layout. This level represents the core of RAISO because it describes the main components over which a control table of an interlocking system is derived. The terminology adopted to describe the concepts is compatible with RailML. A railway track layout is represented by class *Interlocking_Area* which has been defined as restrictions of object property *DUL* : *hasComponenent* on class *ssn* : *SensingDevice* and *Railway_TrackLayout_Node*. This layer includes temporal concepts permitting to observe the state of elements (e.g., occupied, moving, etc.) In RAISO they have been used to represent time instants and intervals. Time instants are represented using the *time* : *Instant* class and the *time* : *inXSDDateTime* property. A *Track_Section* (See Figure 3.5) is declared as subclass of *Railway_TrackLayout_Node* and *ssn* : *Platform*. This class is characterized by starting and ending at a *Railway_Track_Layout_Junction* via the object properties *begins_at* and *finishes_at*. These properties are respectively completed with the following inverse *beginning_for* and *ending_for*.

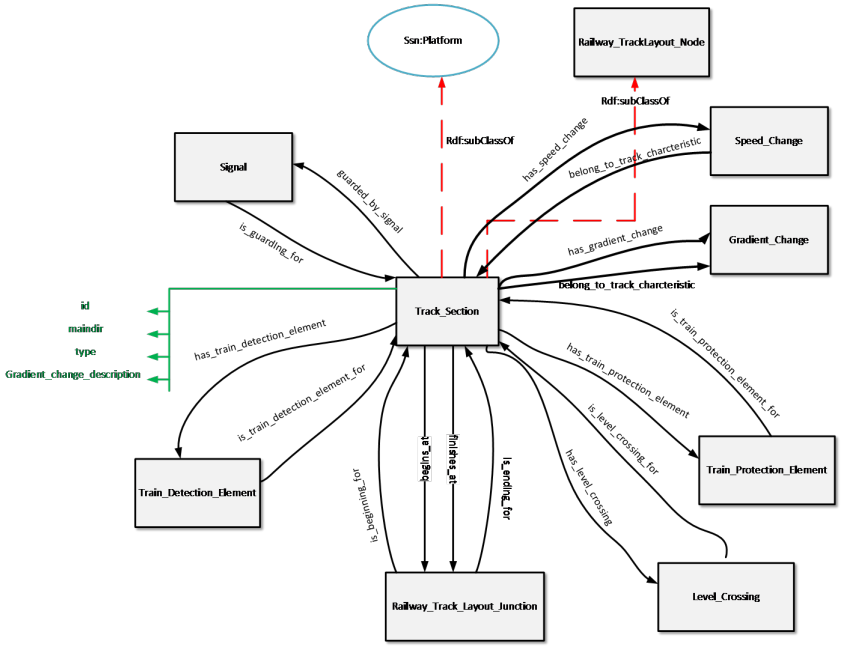


Figure 3.5: TrackSection class and subclasses

Railway_Track_Layout_Junction is a specialization of *otn* : *Railway_Element_Junction* enriched with a general axiom, which is clarified later when discussing about switches. A *Track_Section* has also associated instances of subclass of *ssn* : *SensingDevice* class *Train_detection_Element*, *Train_Protection_Element* and *Level_Crossing* which represent the passive part of signaling. Speed and gradient information of *Track_Section* individuals are obtained by object properties restriction on class *Speed_Change* and *Gradient_Change*. *Train_Detection_Element* (see Figure 3.6)is defined as a subclass of *ssn* : *Sensing_Device* and object property restriction on *TrainDectionState* which permits to assign and observe states of the device (i.e. occupied or not).

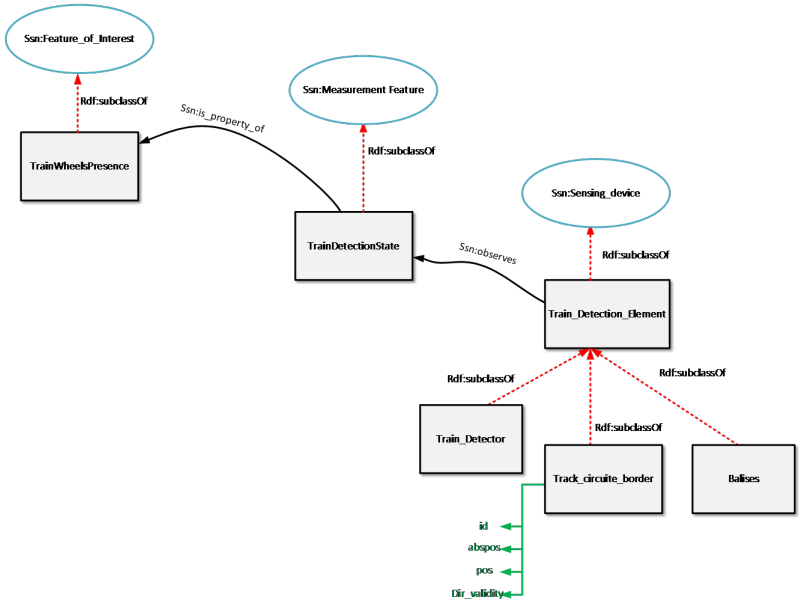


Figure 3.6: TrackSection class and subclasses

$TrainDetectionState$ is defined as a subclass of $ssn : MeasumentOfFeature$ which is related to $TrainDetectionFeature$ by the object property $ssn : is\_property\_of$. $Train\_Detection\_Element$ is related with $Track\_Section$ via the object property $has\_train\_detection\_Element$ defined as sub property of $ssn : attachedSystem$ completed with inverse $is\_an\_Train\_Detection\_element\_for$. A subclass of $Train\_Detection\_Element$ are $Train\_Detector$, $Track\_Circuit\_Border$ and $Balise$.

$Train\_Protection\_Element$ is defined as a subclass of $ssn : SensingDevice$ and object property restriction on $TrainProtectionState$ which permits to assign and observe states of the device (i.e.engaged or not). It is related with $Track\_Section$ via the object property $has\_train\_protection\_Element$ defined as sub property of $ssn : attachedSystem$ completed with inverse $is\_an\_Train\_Protection\_element\_for$. $Level\_Crossing$ is defined as a subclass of $ssn : Sensing\_Device$ and object property restriction on $LevelState$ which permits to assign and observe states of the device (i.e. levelUp, levelDown). $LevelState$ is defined as a subclass of $ssn : MeasumentOfFeature$ which is related to $LevelFeature$ by the object property $ssn : is\_property\_of$. $Level\_Crossing$ is related with $Track\_Section$ via the object property $has\_level\_crossing$

defined as sub property of $ssn : attachedSystem$ completed with inverse $is\_level\_crossing\_for$.

Another extension of $Railway\_Track\_Layout\_Node$ is represented by the class $Switch$ which is also defined as a subclass of $ssn : SensingDevice$ and object property restriction on $SwitchState$. This class permits to assign and observe states of the device (i.e., locked or not). Class $Switch$ subclasses are $Derailer$, $OrdinarySwitch$ and $ThreewaySwitch$. $OrdinarySwitch$ and $ThreewaySwitch$ class are defined respectively as restriction on object property restriction on $Railway\_Track\_Layout\_Junction$ by $begins\_at$ and $finishes\_at$. Active signaling elements are subclass of $Signal$ defined as $otn : Node$. A signal could be fixed as speed signal or milepost indication respectively defined with class $Railway\_Speed\_Signal$ and $MilePost$ or a two-aspect or three-aspect lamp represented by $Railway\_Sempahore$ class. $Railway\_Semaphore$ class is a $Signal$ defined as a subclass of $ssn : SensingDevice$ and object property restriction on $SignalAspect$, which permits to assign and observe states of the device (e.g. showing red, yellow or green).

The most important data property is signal direction, which determines trains movement direction allowed on interlocking area. Further subclasses of $Railway\_Semaphore$ are $Advenced\_Signal$, $Distant\_Signal$, $Home\_Signal$ and $Starter\_Signal$ which are defined as restriction on data property signal function. Routes are conceptualized extending $otn : Route$ with $Railway\_Route$ which is a subclass of $otn : Railways$. For each layer we defined different class of routes. At network layer, a route is represented by $Railway\_Line\_Route$ defined as extension of $Railway\_Route$ and object restriction $otn : hasComponent$ on class $Railway\_Line$. At station layer, a route is conceptualized by the class $Railway\_Element\_Route$ which is defined as a subclass of $Railway\_Route$ and restriction of object property $otn : has\_component$ on class $Railway\_TrackLayout\_Node$ and on has_entry_signal and has_exit_signal on class $Signal$.

3.5 Validation and case studio

In this section, we show how is simple to search relevant information for safety properties defining some consistency and completeness properties examples of the representation. We choose SPARQL as formal language.

3.5.1 Safety Properties

Safety properties verification is strictly depends on the capability to identify track sections and switches per route. The query coded in 3.1 resolves this issue using "property path" SPARQL 1.1 feature, which permits to exploit all possible routes through a graph between two graph nodes.

Listing 3.1: Query 1

```
select ?n1 ?mid ?n2
where{
  ?n1 a raiso:Track_Section.
  ?mid a raiso:Railway_TrackLayout_Node.
    ?n2 a raiso:Track_Section.

  ?n1 raiso:is_connected_to+ ?mid.
  ?mid raiso:is_connected_to+ ?n2.

    ?n2 ssn:attachedSystem ?sn2.
    ?n1 ssn:attachedSystem ?sn1.

        ?sn2 a raiso:Signal;
        raiso:signal_dir ?dir.
        ?sn1 a raiso:Signal;
          raiso:signal_dir ?dir.

  filter (?n1 != ?n2).
  filter (?n1 != ?mid).
  filter (?n2 !=?mid).
  filter not exist (?n1:is_parallel_to ?n2)

}
order by ?n1 ?mid ?n2
```

3.5.2 Consistency

A railway track layout ontological model is consistent if:

1. All individuals belonging to Railway_TrackLayout_Node class and related to an individual belonging to Railway_Track_Layout_Route class

are connected and form a path.

2. Each track section has installed the correct train detection element related to the railway line which it belongs to.

Query 3.2 verifies 1) searching for illegal routes formed by a not connected path. The verification of 2) is determined by examining the railway line containing Track_Section type individuals.

Listing 3.2: Query 2

```
select distinct ?route ?nd1 ?nd2 where {
  ?route a raiso:Railway_Track_Layout_route;
         OTN:contain ?nd1.
  ?nd1 a raiso:Railway_TrackLayout_Node.
  filter not exists{
    ?nd1 raiso:is_connected_to ?nd2.
    ?nd2 a raiso:Railway_TrackLayout_Node.
}
}
```

Query 3.3 simply searches for a high-speed railway line type containing those individual. If the inferred result compared with the actual diverge than the model is inconsistent.

Listing 3.3: Query 3

```
SELECT DISTINCT ?rl
WHERE{
  ?rl a Railwayl_Line_HS;
     otn:contain ?ts.
  ?ts a: Track_Section

}
```

3.5.3 Completeness

A railway track layout ontological model is complete if:

1. Each track section of Interlocking_Area has installed at least one train detection element

2. Each track section of Interlocking_Area has at most two signals must be present between two consecutive track section, installed only at track begin or track end.

Query 3.4 verifies the properties at the same time: we search for the presence of train detection element for each track section and control that each track section has at most two signals that are not overlapped.

Listing 3.4: Query 4

```
ï»¿select ?ts_error where {

{
  ?ts_error a raiso:Track_Section;
            ssn:attachedSystem ?s1;
            ssn:attachedSystem ?s2;
            ssn:attachedSystem ?s3.
  ?s1 a raiso:Signal;
        raiso:signal_abspos ?abspos.
  ?s2 a raiso:Signal;
        raiso:signal_abspos ?abspos.
  ?s3 a raiso:Signal;
        raiso:signal_abspos ?abspos.
}

union {
        ?ts_error a raiso:Track_Section.
   filter not exists {
     ?t a raiso:Train_Detection_Element.
     ?ts_error ssn:attachedSystem ?t.
        }
      }
}
```

As real case it was chosen to analyze Diecimo Pescagli station which is a typical Italian pass-through station (see Figure 3.7) and the Dutch Santpoor Noord Station (see Figure 3.7), which is mentioned in various work [39] [71] [96] [38].

In the first phase, owl model for each station have been constructed: for the Italian station, a new RailML XML file have been compiled while for the

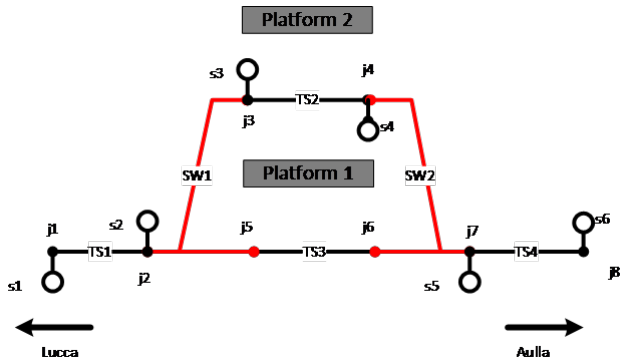


Figure 3.7: Decima Pescagli station

Dutch Station the RailML was gathered from [39]; the two files were then translated to RDF triples using a developed XLST based script. In the next phase, routes information have been generated by applying Query 3.4.

The resulting data has inserted in the RDF store. The last phase consists in testing consistency and completeness properties that were conducted injecting some errors and verifying our detection capabilities. In Table 3.1 a summary for each station is reported.

S	RDF	R	RD	NJ	ND
Decimo Pescagli	367	12	12	21	21
Santpoor Noord	961	36	36	58	58

Table 3.1: Result table

(S) Station
(RDF) RDF
(R) Route
(RD) Route detected
(NJ) Num. of errors injected
(ND) Num. of errors detected

While for data validation, encouraging results have been obtained, errors have occurred during evaluation of a preliminary version of Query 3.1 due a presence of cycles in the RDF graph. To tackle the problem the track sections that are delimited by two switches (e.g., TS2 and TS3 in Figure 3.7) have required filtering.

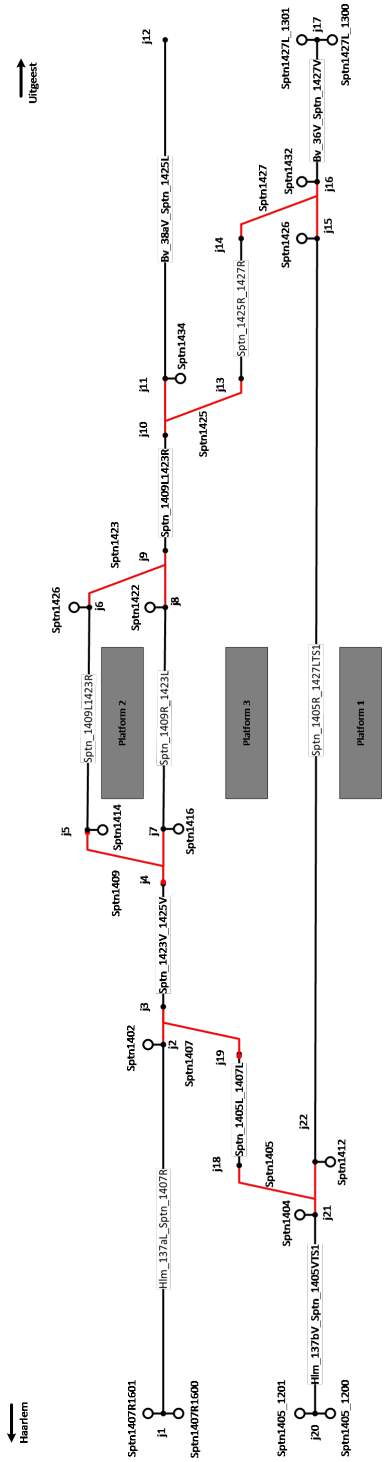


Figure 3.8: Santpoor Noord station

3.6 Conclusion

Railway knowledge could be exploited by non-natural language such as XML language which enables data interoperability. In literature review has been presented some works covering the introduction of complete tassonomy such as RailML and ontology modelling for Centralized Traffic Control (CTC) logics or optimization in terms of network throughput of a new railway infrastructure.

The proposed Raiso ontology covers the railway signalling aspects which is novelty in this ontology railway application. Raiso has been checked against consistency and essential safety properties has been expressed. Once facts have been entered into Knowledge-base grounded on Raiso, it becomes possible to use the relationships contained within the model to infer new information: if th railway engineering state that an object "$sensor_xyz$" is an instance of $Track\_Circuit\_Border$ and the reasoner then inferring that must "$sensor_xyz$" be a $Train\_Detection\_Element$. This is inferred because $Track\_Circuit\_Border$ is a type of $Train\_Detection\_Element$.

Over and above this type of simple common-sense reasoning, Raiso can be further enhanced by the additional of rule-based axioms. Rules make it possible to capture and use more expressive logical statements that are needed for complex decision making, such as the following statement: "if train x is entering track section y at then the signal protecting track section y shows red light" This can be used to infer new facts such as " train x is entering track section y" against the model where relevant. By expressing these rules in the ontology, the operational logic associated with a domain can be stored in the data model, rather than in the code of individual applications, making change management and future development easier.

Part II

Social Media Data Analysis Tools

Part II - Introduction

In recent years, social media have become an important communication tool and instrument for monitoring preferences of users, as well as making predictions in a number of contexts. Many social media platforms allow us rapid information diffusion, and thus may be used as a source of information for early warning and, more generally, for informing and/or advertising many users. Among the various platforms, Twitter may represent the whole category of real time microblogging tools, even if almost each of them has some specific mechanisms.

Twitter.com has a very large user base, consisting of 1.3 billion of accounts and hundreds of millions of users per month. Twitter users can produce a post (i.e., a "tweet"), about any topic within the 140-characters limit and can follow other users to receive their tweets/posts on their own twitter web page, as well as on the mobile App.

Twitter plays an important role for spreading information, allowing people to communicate and share contents with each other in a fast manner. The posts made by a user are displayed on his profile page, and are also brought to the attention of all his/her followers. It is also possible to send some direct private message to other users without provoking diffusion. Another solution is to create a tweet including in the 140 character a direct mention of a user by using a prefix @ such as "@usernickname".

In this case the @usernickname is stimulated by receiving a notification. Therefore, information in the tweets is diffused over the social network users through retweets of the former tweet, thus echoing the early message to the followers, thus producing a chain of messages since the retweets are also echoed. When one receives a retweet, it represents the echo of an original tweet made by one user that has been automatically forwarded by Twitter.com to him/her since in principal he/she is following the retweeting user

(a part for eventual promotions performed by Twitter.com for featuring the most important tweets when they are getting on the list of the most appreciated).

Chapter 4

Data Mining the human sensors

This chapter presents Social Data mining. In particular in section 4.2 an introduction to social media analyses is given, in section 4.2 state of the art is shown and in section 4.3 functional requirements are drawn.

4.1 Introduction

In recent years, the usage and popularity of social media (SM) has significantly increased. Meanwhile, a new role for human users is emerging, acting as smart sensors in the ubiquitous web and sharing observations and contents using mobile devices and the web, collecting, analyzing and disseminating information through more and more connected sensor networks. Actually, customers and users are increasingly used to: write opinions on different topics; publish reviews for sharing feedbacks about products, services or market trends; share information on social networks such as Twitter or Facebook.

Therefore, a huge amount of unstructured and publicly available information is daily produced online, which is a matter of great interest both for research and commercial purposes. The most popular social network platforms, with their large-scale repositories of user-generated content, can provide several kinds of insights into the emotional feels of the global online community [91]. Social media analysis is the practice of gathering and

analyzing data to gain new insights which could help to make informed decisions. Specifically, Social Media Analytics (SMA) is quite an emerging interdisciplinary research topic with the effort of combining, extending and adapting techniques and methods for social media data analysis, monitoring and visualization [73]. Among the several social media solutions, Twitter is one of the most popular microblogging platform (counting about 300 million active users per month [41]), allowing users to have a personal news feed and followers attached to it.

Twitter has emerged as one of the most widespread environment for social media analytics [24], with over 3 billion tweets and 15 billion API calls generated daily. Followers receive notifications connected to the actions performed by the users they follow. Typical actions of users can be: posting a message (tweet), commenting, expressing like/favorite, retweeting (the echo of some tweets by some users to the followers of the retweeting user). Therefore, tweets and retweets are exposed to other Twitter users, thus enhancing the chance of provoking their interests and reactions. Some of these mechanisms can generate viral processes that may lead to a huge diffusion of tweets in the user community.

4.2 State of art

In the state of the art, numerous commercial and research tools for social media analysis have been presented. In particular, for analyzing Twitter data we have: Keyhole Social Media Analytics [14], Tweetreach [19], Brandwatch [10], Followewonk [11], TweetTracker [18], Twitris [20], OSOME [49] and SAS Sentiment Analysis [17]. They are examples of social media analytics platforms, which in some cases measure demographics, influential topics and sentiments.

Keyhole social media analytics [14] targets Twitter and Instagram and offers three services: text annotations tracking (keyword, URL or hashtags), account tracking and historical data retrieval. The text tracking service shows aggregate statistics such as the post volume containing text (in a three days' window), tweets sample, hashtags and word cloud, the number of users involved in the matching tweets, percentage of tweets, replies and retweet, impressions, percentage of sentiment score, percentage of device sources, demographic percentage. The account tracking service shows aggregate statistics as the number of total posts, followers, followees and the

average counts of likes and retweets.

Tweetreach [19] targets Twitter and offers aggregate statistics such as volume of posts, tweet and retweet replies pie charts, top authors, retweet samples and timeline limited to 1.5M tweets and up to the 7 past days. Tweetreach offers also reach and exposure metrics. For instance, Reach metric is defined as the maximum number of unique Twitter accounts that received tweets containing keywords, hashtags or user citations specified in the search query during the 7-days limited time window; while Exposure metric is the total number of times tweets containing the search query keyterms were delivered to Twitter streams.

Brandwatch Analytics [10] is a web-based social media monitoring platform designed to allow users to get the most out of the social media data relevant to their business. It is focused on providing a good customer experience. Its main use cases are: brand/reputation management, finding influences/advocates, market research, campaign, crisis management, community management, customer services, SEO and lead generation. Channels feature allows tracking any public Facebook page or Twitter account without the need for administrator rights.

Followewonk [11]as the name suggests, is a Twitter analytics tool user network analysis centered. Followewonk has five modules Search Twitter Bios, Compare Users, Analyze Followers, Track Followers and Sort Followers. The search feature allows to retrieve the Twitter users involved in the key terms contained in the search query. The compare users feature allows comparing Twitter users by followers. The analysis of followers allows to get information about followers such as mapped locations, most active hours, inferred gender, counts, etc. The Track Followers feature displays an interactive graph tracking changes in user gained and lost followers over certain time periods. The Sort Followers feature allows to sort all Twitter users' followers. It is possible to add a second filter to whichever list user is analyzing and see temporal trends on Twitter, tweet counts, number of followers, number of followers and social authority.

Twitris [92] is a social platform project developed at Kno.e.sis which is tailored on specific events such politics and elections, social movements and uprisings, crisis and disasters, entertainment, and environment. The main dashboard shows a map overlaid with markers indicating the spatial locations from which tweets were gathered for each event loaded in the platform. Moreover, each marker is enriched with related news, sentiment score, multi-

media (images and videos) and Wikipedia articles by using explicit semantic information from DBPedia and SPARQL over metadata extracted from the tweets.

OSoMe [49] is a multi-modular architecture accessible through : web interface and API. The first group encompasses tools producing time series plots of the number of tweets (defined by one or multiple queries specified as one or multiple terms), tools visualizing how a given hashtag spreads through the social network via retweets and mentions, and tools enabling the exploration of information diffusion through geographic space and time. The available API takes as input a time interval and a list of tokens (hashtags and/or usernames) and returns the following data in the same time interval: a list of tweet IDs mentioning at least one of the inputs, a count of the number of tweets mentioning each input token, a count of tweets matching any of the input tokens, a list of user IDs mentioning any of the tokens and a count of matching tweets produced by each user.

SAS [17] is a general data source analytics platform which gathers any online data, for instance Facebook and Twitter, blogs, and review sites (e.g., TripAdvisor and Priceline). The tool performs sentiment analysis in real-time as the data is being retrieved, generates related reports with given locations where the topic is being discussed, and quantifies perceptions as positive, negative, and neutral. The provided graphic interface permits developing and managing sentiment analysis models (e.g., create, refine, upload). The SAS tool provides a workbench functionality through APIs, allowing automatic/programmatic integration with other modules/projects.

In addition to the above-mentioned solutions, Sentiment Analysis of social media is typically used for assessing consumer feelings [94], predicting financial and market results [37], predicting election outcomes [93], providing early detection and warning for adverse medical issues [23], as well as for disaster response surveillance systems [60]. Research techniques and solutions in the field of Sentiment Analysis are divided into three main approaches [44]:

- Unsupervised learning: these methods relying usually in creating a sentiment lexicon in an unsupervised way, and using NLP linguistic-based rules to estimate the sentiment polarity.
- Supervised learning: input text is modeled as a feature vector representing its main characteristics to be estimated by mean of machine learning algorithms, typically using classifiers trained on collections of a priori annotated or labelled corpora.

- Concept-level Sentiment Analysis: these solutions are applied at concept level instead of word level. Trying to extract multi-word expressions from text, which are useful for sentiment analysis since they carry specific semantics and sentics [42].

4.3 Functional Requirements

Effective social media measurement and assessment requires a combination of metrics derived by cross-domain techniques, and several different metrics should be combined to extract relevant information [90]. Besides, the complexity of analysis may significantly vary, per different business or research contexts.

In a Twitter data analytics platform, the visualization must permit data manipulation and filtering in an interactive way, to quickly identify anomalies and outliers. Moreover, it must be able to visualize streaming data at different time resolutions (per years, months, weeks, days, hourly, in real time). Processed data must permit also predictive analytics supporting decision processes, such as regression, predictions, clustering, machine learning. There must be the possibility to extract processed data from the platform, so that they can be processed later independently to identify how they work on analyzing, predicting or early warning of events based on social media data. A social media analytics platform must deal with network analysis. The tool must be able to identify engagement and influence. [90] also proposes some specific use cases and requirements to be considered, such as the need to implement efficient methods to identify new or emerging relevant topics or themes to be analyzed, as well as for the identification of specific segments of social media audience as potential targets. Twitter analytics platforms must deal with tasks such as Twitter data retrieval (e.g., tweets, users' info, etc.), store crawled data, implements efficiently text data analysis (e.g., Natural Language Processing, Sentiment analysis) and network analysis algorithms. Although social media analytics platform stores a minimal subset of tweets, every second, on average, around 6,000 tweets are tweeted on Twitter which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year [12], therefore a social media analytic platform must deal with large data store. The social media analytics platform must adopt convenient computation strategies according to the performed task. Building the network of retweets for a given hashtag will

take more time and computational resources than just counting the number of posts containing the hashtags. Moreover, it should be considered as a relevant aspect the fact that tools available for researchers are far from being ideal. They are usually provided with uncomplete access to raw data without analytic tools, which have to be designed and implemented from scratch [30]. Twitter provides many different modalities to access Twitter data, among which the most efficient for recovering tweets are the Search API and the Streaming API. Since version 1.1 of Twitter API, it is necessary to log into Twitter by using OAuth protocol for all requests. Both Twitter APIs types return data in JSON format. Search API presents a limited number of requests every 15 minutes. The Streaming APIs give developers a low latency access to Twitter's global stream, but limited access to the whole tweets. Twitter offers different streaming endpoints customized for use type: public, user and site. Both the Search and the Streaming APIs present some limitations in terms of maximum number of tweets per hour, and any of them do not guarantee that all tweets which are on Twitter.com could be obtained for the analysis. For example, The Search API is not offering a complete index of all tweets, but instead an index of recent Tweets (published in the past 7 days). Total rate limits of Streaming API are quantitatively not well documented (the documentation say up to 1% of the full firehose of tweets). Therefore, none of the two types of APIs allows a complete recovery of the entire Twitter corpus for a given topic and/or search key. Furthermore, the maximum number of request calls the Twitter APIs can handle in a given period is limited (Twitter limits each API call to collect at most 100 tweets per request). This fixed threshold proves to be not suitable, since the 100 tweets limit per request seems to be well balanced when monitoring long-term events, hence compromising the capability of recovering all tweets and retweets for very fact events, producing high number of tweets/retweets per second.

Other important factors, such as the huge number of tweets that can be produced and collected for certain cases, the complexity of social relationships among users, the limited size of tweets (140 characters), and the fact that historical Twitter data are not accessible via the Twitter APIs force the developers to set up specific architectures for collecting tweets, while attempting to retrieve them with sufficient reliability [83].

Furthermore, companies such as Twitter are both restricting free access to their data and licensing their data to commercial data resellers, such as

Gnip and DataSift [30]. On the other hand, when monitoring explosive/fast events (typically with a high volume rate, in the order of hundreds thousands of tweets per day/hour), the same limit is too restrictive, thus affecting the precision of tweets gathering. It is worth noting that such an aspect is often underestimated by social media analytics tools at the state of the art, or at least no guarantee or realistic measure on their effectiveness is provided in this sense. In addition, as mentioned before, none of the Twitter APIs may guarantee the full access to the tweets.

According to the above analysis of the state of the art, a set of requirements have been identified and thus a feature comparison among the reviewed solutions and the proposed Twitter Vigilance framework is showed in Table 4.1. The features used for the comparison are defined in the following:

- Twitter Metrics: insights about tweets volume metrics (tweets, retweets over time etc.);
- Sentiment Analysis: techniques and solutions adopted to extract a quantitative measure of users' sentiment from published textual contents;
- Natural Language Processing (NLP): to extract linguistic information from textual contents, useful for further investigations (e.g., Sentiment Analysis);
- API availability: accessibility via API;
- User network analysis and, generally, users' based metrics;
- Data analysis based on geolocation (e.g., geographic clustering);
- Real -time analysis: the capability of the system to perform Twitter data analysis on a real-time basis;
- Full faceted search: in order to have further views of data analyzed on tweets collection it is mandatory the exploitation of full faceted search [99], which enable users to browse tweets by choosing from a pre-determined set of categories (tweet author, message, url, geolocation etc.); Metrics for quantitative assessing the efficiency of message recovery, which is an important aspect to assess the system capability of recovering all the available tweets, due to the several limitations imposed by the Twitter APIs;

- Minimization of searches on Twitter: query optimization is a critical part in the functional flow of a Twitter analytics platform. Query parser must take account of Twitter API limit and issue an evaluation plan to minimize twitter API request.

Service	TM	SA	NLP	API	UN	GEO	RT	FS	ME
SAS	N	Y	N	Y	N	N	Y	N	N
Keyhole	Aggregate	N	N	N	Aggregate	Y	N	N	N
Tweetreach	Aggregate	N	N	N	Aggregate	Y	N	N	N
Brandwatch	N	N	N	N	Y	Y	Y	N	N
Followewonk	N	N	N	N	Y	Y	Y	N	N
Tweetris	N	Y	N	N	N	Y	Y	N	N
Osome	Y	N	N	Y	Y	Y	Y	N	N

TM Twitter Metrics (e.g. number of tweets, retweets over time)

SA Sentiment analysis

NLP Natural Language Processing analysis

API API availability

UN User Network analysis

GEO Data analysis based on geolocation

RT Real time Analytics

FS Full faced Search Queries

ME Metrics for assessing recall efficiency

Table 4.1: System Efficiency for Original Tweets Coverage and Twitter Api Call Saturation Levels on Overall Message Retrieval

Chapter 5

Twitter Vigilance

In this chapter we describe a system for retrieve and analyse tweeter streams. In particular in section 5.1 a data analytical tool developed and maintained at DISIT Lab is presented, in section 5.2 the system architecture is discussed , in section 5.3 two case studio are shown and in section 5.4. [1]

5.1 Introduction

The Twitter Vigilance platform [21] has been designed as a multipurpose comprehensive tool providing different tasks and metrics suitable for Twitter Search API, with the aim of Twitter data collection, analysis and monitoring for research purpose. Search API has been chosen, since it offers a greater flexibility and chance of filtering query results, given the number of parameters it provides. The architecture is depicted in 5.1.

[1] The part of this chapter related to system architecture and applications has been published as "Proceedings" in *(GARR_16) Twitter Vigilance: Modelli e Strumenti per l'Analisi e lo Studio di Dati Social Media ed il Monitoraggio in Real Time", pp 24-26 Conferenza GARR_16 ISBN 978-88-905077-6-2* and in (2017 IEEE 3rd International Smart World Congress) "Twitter Vigilance: a Multi-User platform for Cross-Domain Twitter Data Analytics, NLP and Sentiment Analysis"

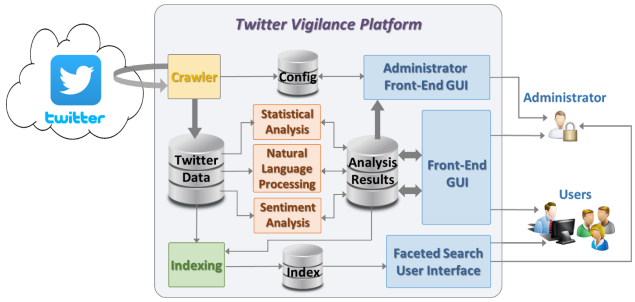


Figure 5.1: Twitter Vigilance architecture

In Twitter Vigilance, a distributed crawler performs data gathering and extraction by using Twitter Search API. The data acquisition approach is based on the concept of Twitter Vigilance Channel, consisting in a set of simple and complex search queries which can be defined by a registered user by combining keywords, hashtags, user's IDs, citations, etc., in a structured logical syntax, according to the search syntax of Twitter.

The search queries associated with each Twitter Vigilance Channel are posed to the Twitter platform via a crawler. Both configuration parameters and statistical results are accessible from the front-end interface for the user. Collected tweets are made accessible to the back-office processes, which implement statistical analysis, natural language processing (NLP) and sentiment analysis (based on distributed NLP on Hadoop [7]), as well as general data indexing. The metrics resulted by the back-office processes are stored on a dedicated database and made accessible to the front-end graphic user interface (see Figure 5.2 as an example), which allows visual analytics, temporal trends and time series visualizations, data results navigation, Twitter users statistics and analysis.

All these kinds of analysis are performed at both Twitter Vigilance Channel level and at single search level. In the specific, the following information and metrics can be retrieved: number of tweets and retweets; user citations (to detect potential influencers, pushers, emerging citations, etc.); hashtags (to understand which are the most used, emerging, evolving, etc.); keywords tagged with their part-of-speech (that is, their grammatical function), in terms of nouns, verbs, and adjectives; sentiment analysis; relationships among users; etc. The derived metrics and information can be useful to understand which are the most widely used or emerging hashtags, as well

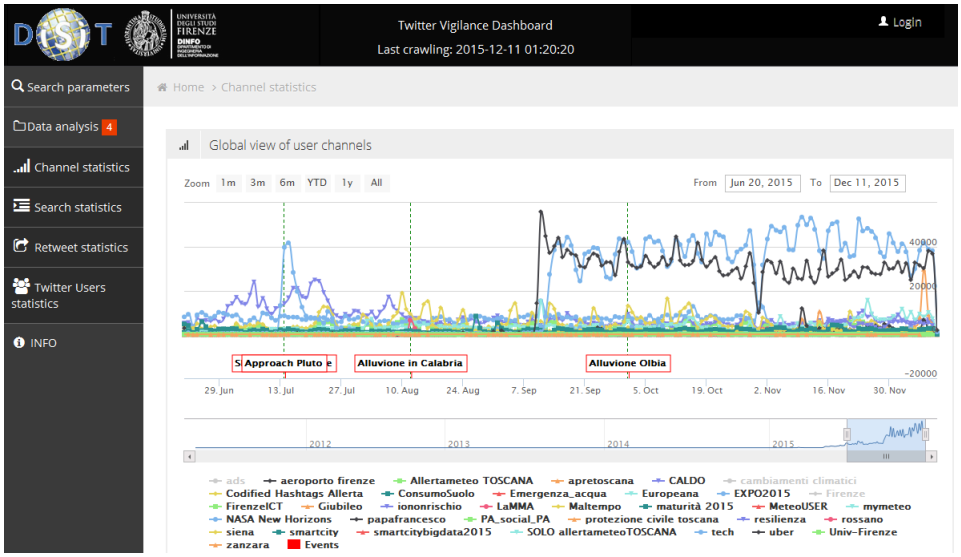


Figure 5.2: Twitter vigilance frontend GUI

to detect which are the most influential in determining the positive/negative signature and polarity detection in the sentiment analysis, and thus for better tuning the tweet collected and for precomputing basic metrics that can be useful for the researcher to make further analysis in different domains and generically for communication and media, predictive models [63], [61]. It can be a useful tool for identifying reasons for positive/negative tweets, as well as the reaction of the community.

5.2 System modules

Twitter vigilance is a modular architecture composed of:

- TV: Twitter Vigilance main tool (http://disit.org/tv/), collecting and analyzing tweets daily;
- RTTV: Real-time twitter Vigilance (http://disit.org/rttv/), collecting and analyzing tweets in real time;
- TVSolr: Twitter Vigilance Advanced search (http://tvsolr.disit.org/), indexing tweets and faceted search.

In the two former Twitter Vigilance tools, a multithread crawler performs data gathering and extraction by using Twitter Search APIs. The data acquisition approach is based on the concept of Twitter Vigilance Channel, consisting of a set of simple and complex search queries, which can be defined by a registered user by combining keywords, hashtags, user's IDs, citations, etc., in a structured logical syntax, according to the search syntax of Twitter. TVSolr is a Twitter Vigilance Advanced search tools which permits to browse efficiently the pool of gathered tweets indexed using a four-shards Apache Solr cluster which allows full faceted search and filtering; the final visualization interface has been realized by using a HUE graphical interface.

Therefore, Twitter Vigilance platform provides a large number of features with respect to other state of the art solutions as depicted in Table I. In addition, it:

- allows registered users (researchers) to create and edit customized channels as a collection of searches on API;
- crawls tweets, computes metrics, and shows results of Twitter Data, as: volume metrics about tweets, retweets and user statistics, NLP and Sentiment Analyses based metrics;
- provides public access to metric results computed on channels and search analysis;
- provides user based metrics such as the number of users per channel/searches, number of posts per users per channel, percentage distribution per search, detailed user information such profile creation, number of favorites tweets, number of followers, number of following, number of list, location, number of tweets retweets;
- Allows the researchers to download resulting metrics values (trough API service) over time for further analysis.

The search queries associated with each Twitter Vigilance Channel are posed to the Twitter platform via a crawler. Both configuration parameters and statistical results are accessible from the front-end interface for the user. Collected tweets are processed by the back-office processes, which implement statistical analysis, natural language processing (NLP) and sentiment analysis (based on distributed NLP on Hadoop [7]), as well as general data indexing. The metrics resulted by the back-office processes are periodically

computed and stored on a dedicated database. This allows to make them accessible to the front-end graphic user interface (see Figure 5.2 as an example) with the needed performance of the front end server.

This allows us customizing the search query, dashboards, reports and file export computed metrics on tweeter data (e.g., to Excel or CSV format) for visual analytics, temporal trends and time series visualizations, data results navigation, Twitter user's statistics and analysis. Specifically designed strategies have been implemented in order to make the whole back office processing more efficient and to maximize consistency of retrieved data, such as: optimized search strategies for collecting tweets (e.g., avoiding repetitions of more inclusive search queries, on the basis of user defined logical rules for searches associated at each channel); incremental processing and caching in order to avoid analysis of search queries which are shared among different channels (typically also on different temporal windows); adaptive real-time processing to optimize the usage of API calls, due to the restrictions and limitations imposed by Twitter previously discussed.

All the aforementioned kinds of analysis are performed at both Twitter Vigilance Channel level and at single search level. In the specific, the following information, metrics and features can be retrieved. Original Tweet Retrieval: Twitter Vigilance takes care of the original tweet of retweets. With the term "original tweet" we define a tweet which has at least one retweet. For each retweet, the original tweet is identified, checked whether these tweets have already been collected or not by the system.

This may happen using the Search API. Twitter search API accepts a maximum of 100 IDs per request. Due to this limit, even though this process is scheduled once a day, it can sometimes occur that not all original tweets are recovered. Thus subsequently, the missing tweets have to be requested to Twitter to have a global view. The request is made through a specifically designed periodic process, which on the basis of the original tweets ID retrieves the missing original tweets. The coverage of original tweets is important in order to identify sources of potential viral events and topics and to make content-based data processing more efficient (for instance, in NLP and Sentiment Analysis related tasks, original tweets analysis is cached so that text content of retweets is not redundantly processed multiple times.

User influence network and other users' metrics Twitter Vigilance performs a user centered statistic and users relationships analysis. The user statistics include the number of users per channel, the number of user per searches, the number of posts per users per channel, the percentage distribution per search, detailed user informations such profile creation, the number of favorites tweets, the number of followers, the number of following, the number of lists, location, the number of tweets and retweets. The relations between users are described as a diffusion network (see Figure 5.3), where nodes represent users and an edge drawn between any two connected nodes indicates an exchange of information between those two users, i.e. the retweets. Edges have a weight to represent the number of messages connecting two nodes. Convenient information such as the number of tweets, followers and followees are present as label near the nodes. Also, a visual graphic semantics is adopted: each node is divided in two parts ranging from light gray to black, which indicate the number of followers and followees, respectively. Darker color indicates high values. To preserve graph clarification, graphical editing features are provided such as min magnitude, node size, hide following or follower.

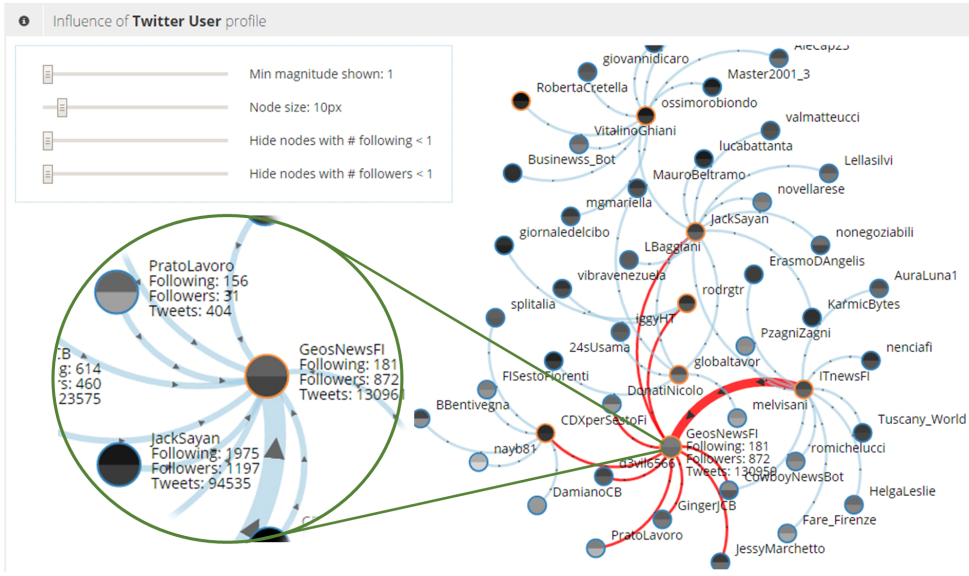


Figure 5.3: A screenshot of a user influence network

Volume based metrics number of tweets (TW) and retweets (RTW) at channel and search level, Aggregated measures at different time resolution (daily, on hourly basis, every 5 minutes for real time processing etc.) are computed and stored on a dedicated database to be quickly accessed and made available for graphical scripts. In addition, higher level metrics are computed, such as the ratio RTW/TW, which provides a measure of users' reactivity to specific topics or events. The ratio RTW/TW may lead to large values, for instance if the event under monitoring becomes strongly viral. NLP-based metrics: extraction of Part-of-Speech (POS) tagged keywords and computation of keywords occurrence at different time resolution (daily, on hourly basis, every 5 minutes for real-time processing on RTTV), grouped by nouns, adjectives, verbs, hashtags and user citations/mentions. This kind of information allows the users to analyze and identify emerging trends or firing/explosive events regarding specific keywords, hashtags (to understand which are the most used, emerging, evolving) or mentions (to detect potential influencers, pushers, emerging citations etc.).

Sentiment Analysis based metrics sentiment polarity extraction for each single tweet; this kind of information can be useful to assess and estimate the general sentiment of the Twitter community regarding a specific channel or search (i.e. regarding specific topics of interest). In addition, the sentiment polarity is extracted at a more fine-grained level, that is for each single noun, adjective and verb. These lower level metrics are used to weight keywords occurrences (computed as NLP-based metrics, as previously described) in order to evaluate the most influential keywords for sentiment analysis, as well as identifying possible sources and reasons to explain or interpret specific sentiment trends. A more detailed description of the Sentiment Analysis is provided later in the section. Aggregated measures of all these are computed at different time resolution (daily, on hourly basis, every 5 minutes for real time processing etc.), and stored on a dedicated database to be quickly accessible and available for graphical scripts. The derived metrics and information can be useful to understand which are the most widely used or emerging hashtags, as well to detect which are the most influential in determining the polarity (the positive/negative signature) of the sentiment, in order to better tuning the collected tweets, and for pre-computing basic metrics that can be useful for the researcher to make further analysis for different domains, communication and media, predictive models etc. It can

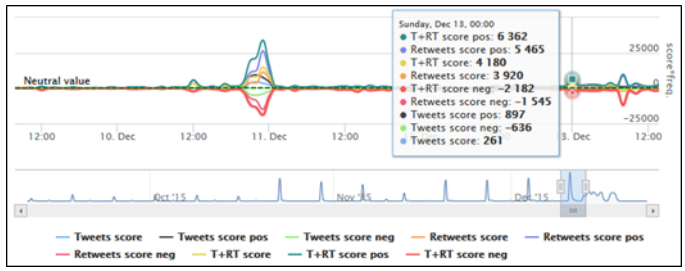
be a useful tool for identifying reasons for positive/negative tweets, as well as the reaction of the community. NLP and Sentiment Analysis related tasks are performed in a multi-language fashion, currently supporting English and Italian. The computation of Sentiment Analysis metrics has been performed by exploiting SentiWordNet [53], a semantic knowledge base specifically designed for Sentiment Analysis. SentiWordNet assigns sentiment scores to each extracted keyword in order to estimate the general sentiment polarity of the collected tweets. In SentiWordNet independent positive, negative, and neutral sentiment values (i.e., real numbers varying in the interval from -1 to 1) are associated with about 117 thousand synsets. In order to perform the analysis in both English and Italian languages, the SentiWordNet lexicon (originally designed in English) has been automatically translated to an Italian version, on the basis of MultiWordNet [85], a resource which aligns WordNet English synsets to the Italian ones. For each single tweet-/retweet, its overall polarity score is given by the sum of all the sentiment weighted keywords extracted in it. A detail of the Sentiment Analysis and NLP manager interface is shown in Figure 5.4.

Recently, Twitter Vigilance has been also made accessible as real time computation of statistical and sentiment analysis for specific dedicated analysis. An example of the channels under observation in real time can be checked at http://www.disit.org/rttv.

5.3 Applications and case studies

In this section, outcomes and performance results of the usage of the Twitter Vigilance platform are reported. As aa qualitative evaluation, Twitter Vigilance provides several different functionalities that outperform the state-of-the-art solutions in terms of features. Exploiting all these features jointly (such as the computation of posts'volume and users' metrics, NLP and Sentiment Analysis, API availability, real-time analysis, full-faceted search and optimized strategies to minimize Twitter API calls) allowed the platform to be used to collect and analyze Twitter data in many different real cases. Some examples are reported in the following.

DISIT Twitter Vigilance platform has been adopted in several studies as a tool to monitor city services, critical events and conditions, user behavior, assessing appreciation of services, city response to events, providing predictive capabilities to support decisional processes. It is also adopted in



(a)

(b)

Figure 5.4: Sentiment Analysis and NLP manager interface: (a) trend of the most relevant sentiment analysis metrics, (b) detail of Top-Sentiment rated Italian adjectives.

smart city projects as Sii-Mobility SCN http://www.sii-mobility.org, REPLICATE and RESOLUTE EC H2020 project http://www.resolute-eu.org, for monitoring as an early warning solution to detect critical conditions (inception of critical events on the city, new reaction on drugs, etc.). Twitter Vigilance currently contains more than 100 channels created by about 30 users which are mainly local Public Administrations and research groups to collect information and monitor different topics and trends related to different domains of interest, such as: traffic and public transportation (e.g.: "TPL", "TPL2", "Uber"), weather monitoring (e.g.:"Maltempo", "Meteo Firenze", "Codified Weather Hashtags", "ARPAT", "LaMMA", "PAMeteoNews"), resilience and early warning for natural disasters and other accidents (e.g.: "PAProtezione Civile", "Protezione Civile Toscana", "Allerta Meteo", "Allerta Meteo Toscana", "Mugnone 2016"), tourism and events in Florence and Tuscany (e.g.: "Turismo Firenze", "Firenze"), monitoring large

attendance public events and TV shows (e.g.: "EXPO2015", "EXPO2015Toscana", "Xfactor9", "Xfactor10", "Pechino Express"), terrorism, adverse drug reaction, etc. In addition, in May 2016 Twitter Vigilance TV and RTTV platforms were used in a joint-venture flood dealing simulation composed of civil protection, fire fighters and other local institutions [16] [62]. For this purpose, the channel "Mugnone 2016" (from the name of a creek that flows in the city) has been created for real-time monitoring. Presently a real-time dashboard for monitoring Twitter data is provided [22].

5.3.1 Italian codified hashtags for weather warning on Twitter

In [62], the "Codified Weather Hashtags" channel has proven to be an effective tool to convey useful information on Twitter, with formal and informal sources regarding weather related events and alerts. The authors collected tweets published within a 30 days period, identified by three codified hashtags (in the three different Italian regions, i.e., Liguria, Tuscany and Piedmont). The aim was to assess whether codified hashtags could represent an effective way to align formal and informal sources of information during weather related emergencies. In this perspective, the use of codified hashtags may potentially improve the performance of systems, for automatic information retrieval and processing during disasters. In [63], the twitter data have been demonstrated to be strongly correlated with respect to the temperature of the hot period in the Tuscany region.

5.3.2 Predicting TV programme Audience by Using Twitter Based Metrics

In [48] some metrics based on Twitter data have been identified and presented in order to predict the audience of scheduled television programmes, or whenever the audience is highly involved as it occurs with reality shows (i.e., X Factor and Pechino Express, in Italy). Experiments have been based on the collected Twitter data by using Twitter Vigilance platform.

The produced predictive models have been validated and assessed in terms of quality, while highlighting the predicting capabilities for the analysed cases, namely X Factor 9, X Factor 10, and Pechino Express. In all such cases, the predictive capability of the produced models according to the identified metrics has been proved. In almost all predictive models, metrics

have been defined as the ratio between the number of retweets and tweets collected and related to the major hashtags of events and they have demonstrated high predictive capabilities in explaining visitors/audience volumes. Also the volume of tweets and the sum of tweets and retweets have confirmed their predictive capabilities. Another interesting predictor can be the number of unique users involved, as well as opinion mining features, such as natural language processing and sentiment analysis related metrics earlier described. The proposed models are based on multi-regressive and/or ridge for short term prediction. On the other hand, other models and approaches have been tested without success, as reported in [48].

5.4 Validation on Collecting Tweets Performance

In this section, the validation of the system in terms of efficiency in getting Twitter data is presented, to put in evidence, its capabilities to recover the correct number of tweets and related metadata, despite the Twitter Search API usage limitations, as well as to adapt itself to changes in the set of recoverable tweets. Therefore, to perform a rigorous assessment, two validation metrics have been defined:

- The coverage percentage of original tweets, Co_{TWO}, provides a qualitative parameter on the system's ability to recover all the available tweets. Being TWO the number of original tweets, and M_{TWO} the number of missing original tweets, the coverage of original tweets Co_{TWO} is given by the ratio of successfully retrieved original tweets to the total of identified original tweets, that is: $Co_{TWO} = \frac{TWO - M_{TWO}}{TWO}$
- The percentage of saturations, S%, indicates the system aptitude to adapt to changes in the number of recoverable tweets, that is a measure of the overall system efficiency in recovering messages. Saturation is defined as the inability of the system to accommodate the growth in the number of messages that are available to a given search: since the twitter Search API limits to 100 tweet IDs per request, saturation occurs when a request through a Twitter Search API call produces 100 new posts of 100 returned. The saturation percentage S% is computed as the ratio of the number of saturations S to the total number of Twitter requests Req_{TW} made by the system crawler: $S_{\%} = \frac{S}{Req_{TW}}$

The assessment has been performed on the whole dataset collected in the last 24 months, since the activation of the platform. The assessed data set consists of about 270 million of Twitter messages (of which about 47% tweets and about 53% retweets) grouped into 102 channels (of which 78 are still active) and a total of 1407 searches (of which 1194 are active). Not active searches are those which are not associated with any active channel. The dataset has been clustered into 7 categories (see Table 5.1) according to the total posts volume, given by the sum of collected tweets and retweets for each channel, in order to identify and analyze possible different system adaptive behaviors and results for increasing volumes of posts. The first subset contains all collected posts belonging to channels with tweets plus retweets volume of less than 10 thousand . The second subset is formed by collected posts from channels with number of tweets plus retweets between 10 and 50 thousand, and so on. Results are shown in Table 5.1, while the performance trend is summarized in Figure 5.5.

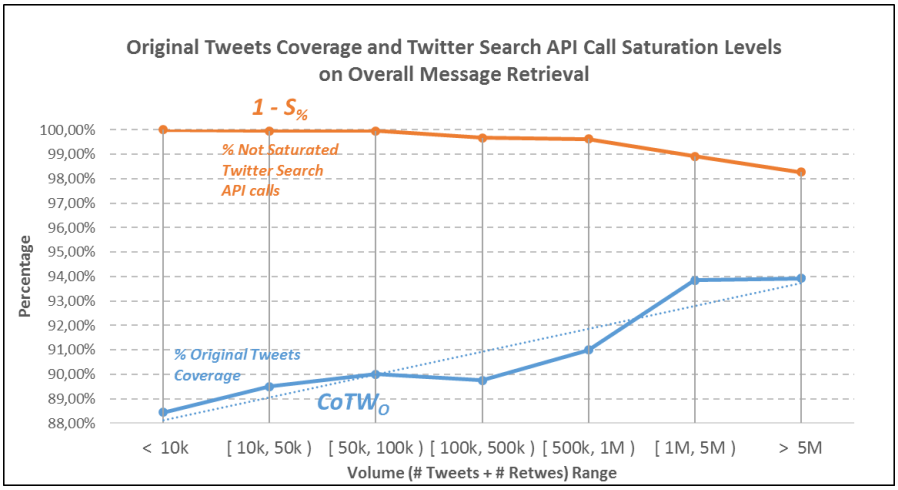


Figure 5.5: Original Tweets coverage and Twitter Search API call saturation percentages on overall message retrieval. The analyzed dataset has been clustered into seven different categories according to the total volume of posts (sum of tweets and retweets).

The system shows very low saturation levels (i.e., very high percentage of not-saturated Twitter Search API calls, as depicted in Figure 5.5) for each

cluster. Results show a very good overall performance of the proposed system in recovering Twitter messages, with an efficiency percentage never dropping below 98%. Saturation percentage slightly increases with increasing number of retweet, as expected, since higher numbers of retweets to be collected may be associated with rapidly variable and/or viral events, which usually lead to a delay in the system adaptation and a consequent higher number of saturations of Twitter requests.

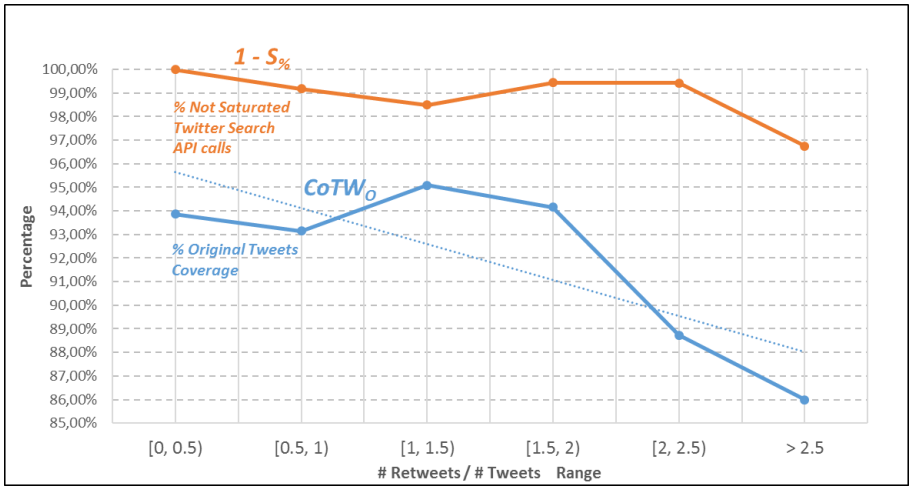


Figure 5.6: Original Tweets coverage and Twitter Search API call saturation percentages on overall message retrieval. The analyzed dataset has been clustered into six different categories according to the ratio of retweets to tweets

Furthermore, a general increasing trend for coverage of original tweets can be observed (for increasing retweet ranges, as outlined by the dotted trend line in Figure 5.5, showing a very good original tweet recover efficiency, from about 89% to 94%. A descending trend should be expected in this case as well. However, the increasing trend can be explained by the presence of the dedicated process for recoverying original tweets. A different behavior is observed when computing the same validation metrics upon a different clusterization of the same test dataset, that is based on the ratio of retweets to tweets. The ratio of retweets to tweets better describes the virality of analyzed events and topics, rather than considering the absolute tweets

and retweets volume. This means that the original tweet recover efficiency decreases with increasing virality.

$T+RT$	# ROT	# MOT	% Co_{TWO}	# SR	# SS	$S_{\%}$	$1-S_{\%}$
$< 10k$	18571	2033	89,05%	124299	1	0,00%	100,00%
$[10k, 50k)$	130051	13716	89,45%	399170	100	0,03%	99,97%
$[50k, 100k)$	96171	10278	89,31%	123804	165	0,13%	99,87%
$[100k, 500k)$	997833	86755	91,31%	849062	1589	0,19%	99,81%
$[500k, 1M)$	930646	61632	93,38%	439956	1998	0,45%	99,55%
$[1M, 5M)$	6454463	439628	93,19%	2787485	31585	1,13%	98,87%
$> 5M$	14714124	899035	93,89%	4509184	64284	1,43%	98,57%

T+RT	Posts Volume (Tweets + Retweets) Range
# ROT	# Recovered Original Tweets
# MOT	# Missing Original Tweets
Co_{TWO}	Original Tweets Coverage
# SR	# Twitter Search API requests
# SS	# Saturations on Twitter Search API requests
$S_{\%}$	% Saturations on Twitter Search API requests (S%)
$1-S_{\%}$	% Not-Saturated Twitter Search API requests

Table 5.1: System Efficiency for Original Tweets Coverage and Twitter Api Call Saturation Levels on Overall Message Retrieval

Chapter 6

DIETA: DIstributEd Twitter vigilAnce analysis

This chapter presents analysys on porting Twitter Vigilance to an distributed architecture. In particular, section 6.1 introduce the problem, section 6.2 presents the tools adopted and section 6.3 gives an overview of the developed artefact.

6.1 Introduction

Every second, on average, around 6,000 tweets are tweeted on Twitter, which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year [12]. When implementing process on these volume of data an efficient - in terms of time and storage consumed - problem may arise (see Figure 6.1).

To tackle this issue two approach exits: scaling up or scaling out.

Scaling up, also known as concurrent programming, is one of the most common methods for utilizing multi-core architecture in the context of a single application. Concurrent programming on multi-core machines is often done through multi-threading and in-process message passing, also known as the Actor Model [68].

The scale-up approach is an example of vertical scalability. Vertical scalability is the ability to increase the capacity of existing hardware or software by adding resources to a physical system for example, adding processing

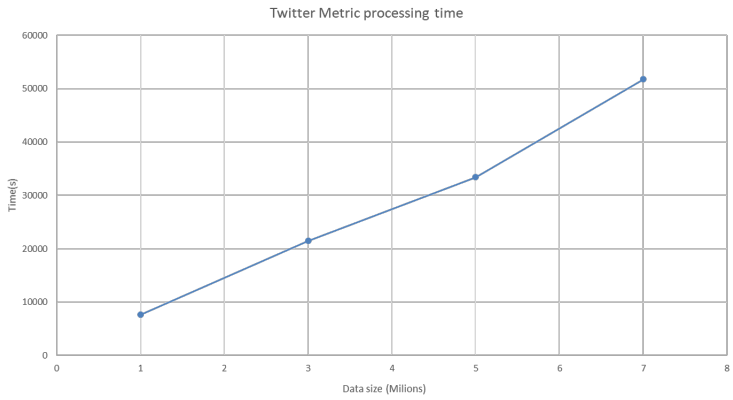


Figure 6.1: Process benchmark on dual core 3.0 GHz 4GB

power to a server to make it faster. In the case of storage systems, it means adding more devices, such as disk drives, to an existing system when more capacity is required.

Conversly, scaling out, also known as distributed programming, does something similar by distributing jobs across machines over the network. There are different patterns associated with this model such as Master/-Worker, Tuple Spaces, BlackBoard, and Map/Reduce.

In terms of hardware resources, scale-out adds multiple, smaller servers running in a distributed computing model, while scale-up adds fewer, more powerful servers that are capable of running larger workloads. In a scale-out system, new hardware can be added and configured as the need arises. When a scale-out system reaches its storage limit, another array can be added to expand the system capacity.

The policy adopted over Twitter vigilance actually is scaling up. The system indeed uses multi-core high performance host to fulfil exposed features.

However, at DISIT Lab it has been studied the porting of Twitter Vigilance to distributed architecture namely DIETA: DIstributEd Twitter vigilAnce.

DIETA is grounded on Apache Big data framework i.e. Hadoop-Hbase stack. Here will be discussed the essential issue and experiments of porting a single host application to distributed one.

6.2 Apache Big data framework

Apache Big data framework is a well-established solution to tackle computational tasks involving big datasets (i.e., on the scale of hundreds of gigabytes, terabytes, or more).

The essential parts of Apache big data ecosystem are : Apache Hadoop and Apache Hbase.

6.2.1 Apache Hadoop

The Hadoop framework is an Java open source project hosted by the Apache Software foundation and it has in Yahoo! one of its biggest contributors. Hadoop implements the MapReduce programming model and it is a free alternative to the Google's MapReduce framework that is not publicly accessible.

Hadoop offers also a distributed filesystem called HDFS that is tightly connected to the rest of the framework. HDFS is not the only choice for the data storage since Hadoop can use other filesystems, like the Amazon S3 for example, but HDFS is heavily integrated with Hadoop and it is easy to install and configure.

HDFS is Master/slave architecture. Daemon services of HDFS are NameNode and DataNode. NameNode maintains and manages the blocks which are present on the Datanode. Data node keeps the physical file metadata information. File breaks into one or more blocks and then stores in a set of Data Nodes whenever user attempts to store very huge size of file [8].

MapReduce is a distributed programming model originally designed and implemented by Google for processing and generating large data sets. The model is built over two simple functions, map and reduce, that are similar to the functions *map* and *reduce* present in functional languages like Lisp.

The Google MapReduce programming model has proved to be performant since the simple principle of mapping and reduce allows a high degree of parallelism with little costs of overhead. Today the Google's MapReduce framework is used inside Google to process data on the order of petabytes on a network of few thousand of computers.

In this programming model all the information is encoded as tuples of the form:

$$< key, value >$$

The work flow of a MapReduce job is this:first, the map function processes the input tuples returning some other intermediary tuple:

$$< key2, value2 >$$

Then the intermediary tuples are grouped together according to their key. After, each group will be processed by the reduce function which will output some new tuples of the form:

$$< key3, value3 >$$

Hadoop Version 2.0 it has introduced a resource management layer (Yarn) to handle different computational frameworks in the same cluster. Yarn can run Hadoop MapReduce, Storm like computational frameworks in the same cluster by sharing resources.

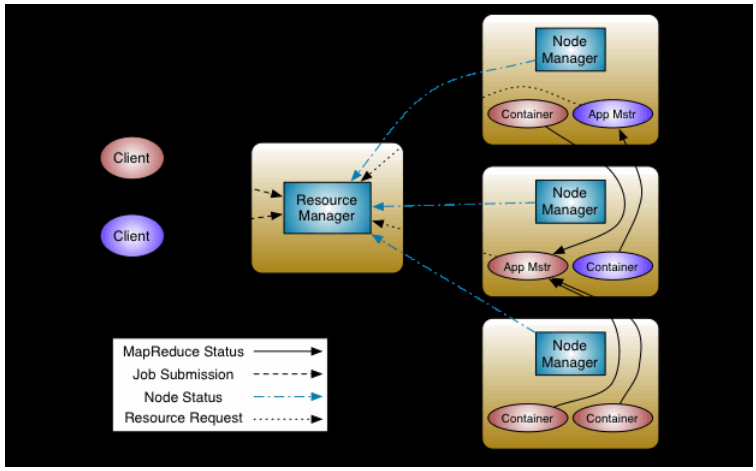


Figure 6.2: Yarn architecture source:apache.org

Yarn has a ResourceManager for scheduling resources of a cluster NodeManagers that runs on computational nodes of the cluster. The ResourceManager is a pure resource scheduler. An application like MapReduce that needs to run on Yarn should implement an ApplicationMaster. The ApplicationMaster is specific to an application and knows how to run the application on the cluster resources. ApplicationMaster asks for resources from the ResoureManager to run its applications. ApplicationMaster can use different

policies for allocating resources and an organization can define their own policies as well. A client program submits an application to the application specific ApplicationMaster along with the resource requirement of the application.

The ApplicationMaster negotiates with the resource manager to acquire the requested resources. The resources are allocated as containers. A container is essentially a set of resources allocated on a node. Now ApplicationMaster can use these containers to run the application on the nodes. To do this ApplicationMaster talks to the NodeManagers that are responsible for the allocated containers and launch the application in these nodes.

After the containers are launched by the NodeManager it monitors them for resource consumption, failures etc and convey this information to the ApplicationMaster.

6.2.2 Apache Hbase

HBase is a master-slave data storage clustered system inspired by Google's BigTable [43]. HBase extends HDFS by allowing real-time random IO operations, whereas HDFS assumes that data will only be written once and always processed in bulk. Data in HBase is organized as a multi-dimensional map. A table is a map of row ids to column families; each column family in turn is a map from columns to values1. HBase has the properties of being distributed (over a Hadoop cluster), scalable, and sparse (i.e., a column only exists in a row if it has a value; there are no NULL columns as in a relational database). These properties have made HBase the tool of choice for companies likeFacebook, Yahoo, and Twitter.

HBase has a basic client API that includes two commonly used features: scans and filters. Scan is used to get data from HBase. It allows us to iterate through a range of rows and lets us limit which columns are returned. Filters are more powerful and effective for working with data. Filters are used to select certain columns or cells based on a variety of conditions. There are many predefined filters provided by HBase but it allows custom filters implementations [9].

HBase uses regions as the basic unit of scalability and load balancing. Contiguous ranges of rows are stored together. Initially there is one region for a table and as it becomes too large, it splits into two and so on. These regions are served by one region server. A region server can have various regions and are in charge of handling all requests for their regions (to add,

remove, etc.). HBase master server is in charge of assigning regions to the region servers for load balancing.

HBase has a coprocessor framework that provides a way to run custom code on a region server. Coprocessors allow us to execute code on a per-region basis, giving trigger-like functionality similar to that of stored procedures. HBase has some classes based on the coprocessor framework which fall into two main groups: Observers and Endpoints. Observers have callback functions (hooks) that are executed when certain events occur, including user-generated, server-internal, and automated events. Endpoints resemble stored procedures where we can invoke them anytime from the client. The endpoint implementation will then be executed at the targeted region or regions and the results from that execution will be returned to the client [9].

6.2.3 Apache Zookeeper

Apache ZooKeeper [13] is an Internet scale distributed coordination system developed at Yahoo and donated to Apache Software Foundation. ZooKeeper provides a centralized service for coordinating distributed applications. ZooKeeper provides few consistency guarantees upon which client applications can be built complex coordinating protocols. ZooKeeper provides the following primitive guarantees to the client applications:

1. Sequential Consistency - client updates are applied on the order they were sent

2. Atomicity -Updates are applied as a whole without any partial results

3. Single System - Client will see the distributed ZooKeeper system as a single system

4. Reliability - Once an update has made, it is persisted

    - The ZooKeeper guarantees that the update is applied and persisted if the client gets a response
    - If an update is seen by a client, that update will not get rolled back in the event of a recovery

5. Timeliness - The client view of the system is guaranteed to be up to date within a certain time bound

With these guarantees, an application can use the ZooKeeper client to build higher-level coordination functions such as leader election, barriers, queues, and read/write revocable locks.

ZooKeeper service stores information in ZNodes. ZNodes are arranged in a tree structure that closely resembles a Unix file system. The difference between a ZNode and a file is that, a ZNode can contain data as well as child nodes unlike a regular Unix file. A ZNode is referenced by a path, which is always expressed as canonical, absolute, slashseparated path similar to an absolute path in Unix file system. Ephemeral node is a special node that only exists until the session that created the node is active. To avoid heavy polling by clients on nodes, ZooKeeper has a concept of watches. A client can set a watch on a ZNode to listen for changes.

6.2.4 Apache Phoenix

While HBase provides a good framework for realtime random access read-/write, there is no well-defined API and comes with no built-in coprocessors. Apache Phoenix's mission is to become the standard means of data access through HBase with the use of a well-defined, industry standard API.

Apache Phoenix is an open source, parallel and relational database layer that is built on top of HBase. It connects to a client-embedded JDBC driver to enable users to create, delete, upsert, and query data, and to provide additional functionality used by most SQL languages. They use their own language that is very similar to SQL.

Phoenix compiles the queries and statements from the client into a series of HBase scans, filters, and coprocessors and then runs them to produce a result set that is retrieved in the order of milliseconds for smaller queries and seconds when using millions of rows.

6.2.5 Ganglia Monitoring System

Ganglia is a scalable distributed monitoring system for high performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It relies on a multicast-based listen/announce protocol to monitor state within clusters and uses a tree of point-to-point connections amongst representative cluster nodes to federate clusters and aggregate their state. It leverages widely used technologies such as XML for data representation,XDR for compact, portable data transport,

and RRDtool for data storage and visualization [78].

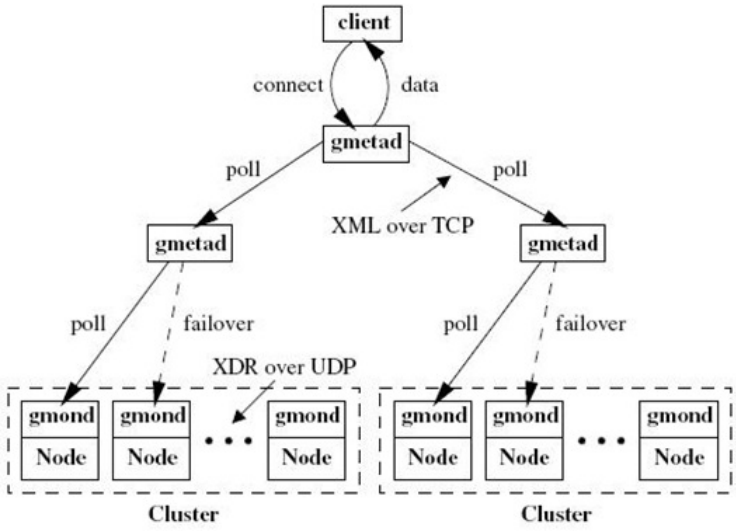


Figure 6.3: Ganglia Architecture

6.2.6 Apache Zeppelin

Apache Zeppelin is a web based and multi-purpose notebook that enables interactive data analytic. The notebook is the place for data ingestion, discovery, analytic, visualization and collaboration. It can make beautiful data driven, interactive and collaborative documents with scala and more. Apache Zeppelin Interpreter allows any language/data-processing-backend to be plugged into Zeppelin JDBC which is Java database connectivity driver provided by Oracle.

6.3 DIETA: DIstributEd Twitter vigilAnce

Dieta as Twitter vigilance is a modular platform grounded on an heterogeneous server (physical and virtual). The Physical server are commodity machine as detailed in Table 6.1.

In figure 6.4 the overall architecture are depicted.

#	Server Type	Ram	# Core	Total Disk
3	Physical	4GB	2	1TB
1	Physical	6GB	2	1TB
1	Physical	8GB	2	1TB
1	VM	3GB	2	1TB
2	VM	8GB	2	700GB
2	VM	6GB	2	700GB

Table 6.1: Nodes detail

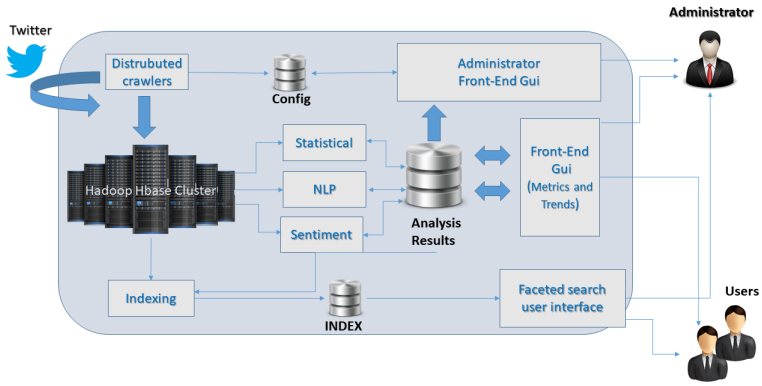


Figure 6.4: Dieta architecture

The modules composing the platform are:

- a module for configuration and computation results
- a module for retrieve and update tweets from tweeter
- a module for enhanced user interface
- a module for indexing
- a module for NLP and Sentiment Analysis

- a module for storage and computation

All module except the last two are the same of Twitter Vigilance tool.

Indexing module is grounded on Lily HBase Indexer which provides the ability to quickly and easily search for any content stored in HBase [15]. It works by acting as an HBase replication sink. As HBase applies inserts, updates, and deletes to HBase table cells, the indexer keeps Solr consistent with the HBase table contents, using standard HBase replication features. The role of the Indexer is to keep the Solr index up to date. For this purpose, the Indexer listens to the HBase Side Effect Processor (SEP) events. This is the listener component in HBase that listens to the changes made in HBase and enables replication asynchronously. The indexer maps Lily records into Solr documents by deciding which records and what fields of the record need to be indexed.

The collected tweets are stored on a Hbase table while analytics process are developed as map reduce jobs. Moreover there are four type of data analytical process:

- tweeting counting process, retweet per day for all or single searches
- tweeting counting process, retweet per day for all or single channels
- calculation of the number of users per day for all or single searches
- process of calculating the number of users per day for all or individual channels

The original process are implemented as sequential php script while the distributed version are Java based. A sample class diagram of this process is depicted in figure 6.5.

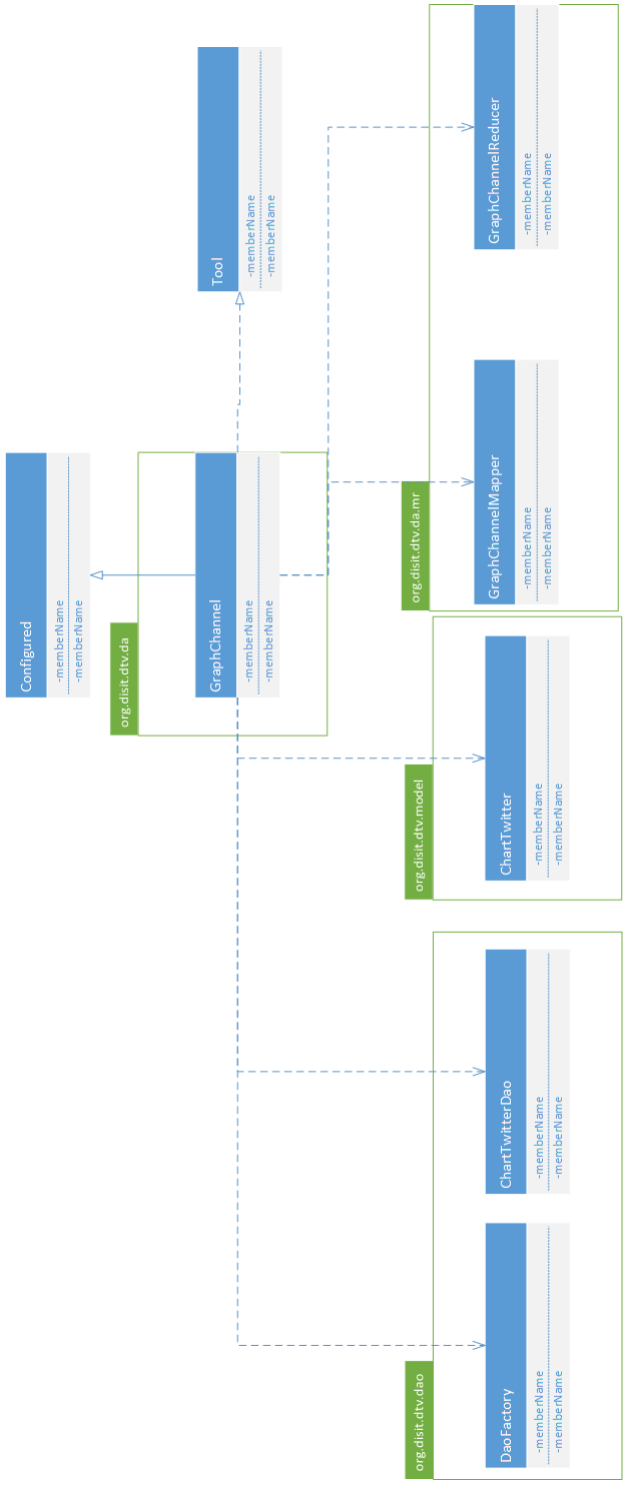


Figure 6.5: Example Map Reduce Graph Processor Class diagram

Finally, Apache Zeppelin is used to permit graphical user interface drill down of data stored in HBASE.

6.3.1 Porting issue

The porting of the single-host calculation process to map reduce facility it is not immediate and may arise problems. The first version of the distributed applications were not efficient indeed.

An evidence is reported in figure 6.6 where the performance of the a generic data analytic process on single host is compared to a process on hadoop-hbase. The size of dataset is reported on the first axis while the time consumed is reported on second axis.

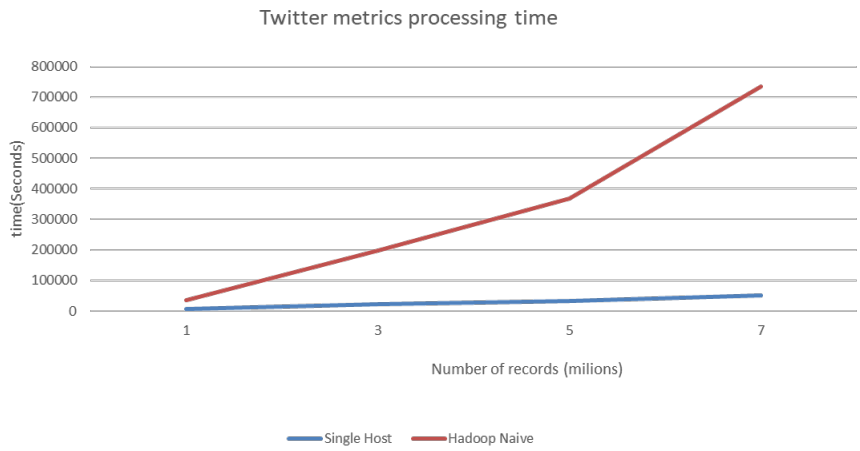


Figure 6.6: Single host vs Dieta Fail

Once the application was developed, it was tested on the a single host server with an instance of twitter vigilance process and on DIETA varying the number of nodes. During the test the cluster status where monitored using Ganglia tool which highlighted weakness as

- under consumption of the available resource
- The application loose too long time **before** starting mapper phase

In algorithm 1 is showed failing process counting the tweets matching a

query string in a dataset and producing a daily frequency for each query stored in the tool.

Algorithm 1 Count number of tweets matching search per day

for all active search s **do**
  $s \leftarrow CreateFilterSearch(s)$
  $TweetCount(s) \leftarrow MapReduceJobCounter(s)$
end for

This algorithm basically translate one-to-one the sequential logic of single-host grounded on optimized SQL queries over indexed table;

problem arise due the fact that the proposed solution does not take into account scheduling time which generally is not less than one minute. Hence if the platform have to compute data for 1000 searches it will require 1000 instance of map-reduce which implies an overall time to process of 1000 minutes.

Conversely in algorithm 2 a parallel-minded solution is showed. Basically the problem is treated as a word-count map reduce applying in a second phase a filtering on the words matching the search.

Algorithm 2 Count number of tweets matching search per day

$dataset' \leftarrow MapReduceJobWordCountPerDay(dataset)$
for all record $\in dataset'$ **do**
  $TweetCount(s) \leftarrow MatchingSearch(s)$
end for

As depicted in figure 6.7 results confirm the performance expected by a Hadoop cluster i.e linear scalability versus number nodes.

6.4 Conclusion

Twitter Vigilance has been designed as a cross-domain, multi-user tool for collecting and analyzing Twitter data. Actually is instanced as a single host running the data analytics's process and the storage engine of database management system.

In this chapter has been presented a study concerning scale out the platform.

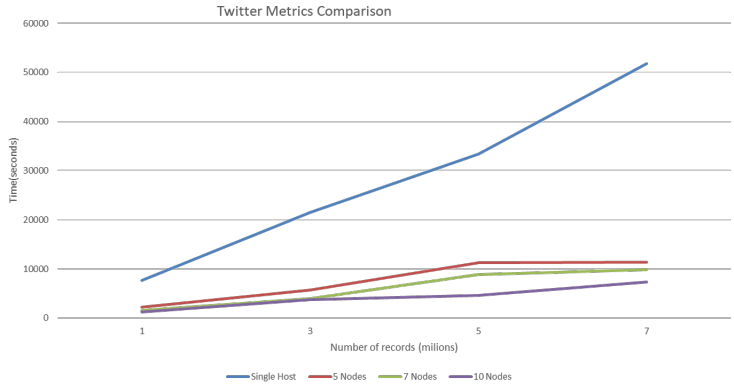


Figure 6.7: Single host vs Dieta

Scaling out main advantage is its ability to scale just by adding "one more node". An additional node or server can be added to the cluster and the necessary modifications to scripts or applications can be quickly implemented.

On the other hand, as has been shown in this chapter, it requires the developing skillets and management capabilities to manage complex Hadoop cluster which require setting up the software on multiple systems, and keeping it tuned and running.

Chapter 7

Conclusion

This chapter summarizes the contribution of the thesis and discusses avenues for future research.

7.1 Summary of contribution

This dissertation summarizes the research work carried out at DISIT lab. (Distributed Data Intelligence and Technology Laboratory) of the Department of Information Engineering (DINFO) at the University of Florence as part of my PhD activity. The work has involved mainly two application contexts. The first one is the software modelling related to railway signalling and control systems. The second, however, refers to technologies for social media analysis.

Rail network uses signals, branch devices and train detectors at the side of the track to control and ensure the safety of the movement of trains from one section of track to another. In particular the system which is the solely involved is the Railway Interlocking System (IXL) which must be compliance with specific European safety standards as EN50126, EN50128 and EN50129.

In the first part it has been discussed the development and the exploitation of model ontology which allows to formalize the layout of a railway network focusing on railway terminal (station) including the position of the track elements (signals, train detector etc...). My investigation consisted in formalizing logical rules, which allow checking completeness, consistency of the model. A case study regarding the analysis of two different stations was

carried out.

The experiments conducted have shown that one's the ontological model is established relevant information and verification procedures are easily achieved by automatically inferring the layout of the remaining installations (i.e., track circuit, signals and track sections) and the possible train routes through the network.

The second part presents social media analysis and related tools. In particular, Twitter has revealed to be one of the most widespread microblogging services for instantly publishing and sharing opinions, feedbacks, ratings etc., contributing in the development of the emerging role of users as sensors. However, due to the huge amount of data to be collected and analyzed and limitations on data access imposed by Twitter public APIs, more efficient requirements are needed for analytics tools, both in terms of data ingestion and processing, as well as for the computation of analysis metrics, to be provided for deeper statistic insights and further investigations.

The study was conducted using Twitter Vigilance platform realized by the DISIT Lab at University of Florence. Moreover, it has been developed the distributed architecture version. The present work illustrates also the challenges and pitfalls of deploying a Big Data analytics system that aims at extracting value from Twitter data.

Part III

Appendix

Appendix A

Acronym

API	API
ATP	Automatic Train Protection
CIS	Computerized Interlocking System
API	Application Programming Interface
CSP	Communicating Sequential Process
DBMS	Database Management System
DISIT	Distributed Systems and Internet Tech lab & Distributed Data Intelligence Lab of UNIFI
DUL	DOLCE Ultra Lite
FOAF	Friend of a friend
GDF	Geographic Data Files
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
IT	information technology

JDBC	Java DataBase Connector
JS	JavaScript
JSON	JavaScript Object Notation, a lightweight data-interchange format easy for humans to read and write and, for machines, to parse and generate
KB	Knowledge Base
N3	Notation3 language, a compact and readable alternative to RDF's XML syntax
NLP	Natural Language Processing
NoSQL	No Structured Query Language
ODP	Ontology Design Pattern
OTN	Ontology on Transportation Network
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SKOS	Simple Knowledge Organisation System
SPARQL	SPARQL Protocol and RDF Query SQL Structured Query Language
SVG	Scalable Vector Graphics
SSN	SemanticSensorNetwork Ontology
SSO	Stimulate-Sensor-Observation Pattern
UI	User Interface
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

VM	Virtual Machine on cloud
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSLT	eXtensible Stylesheet Language Transformations

Appendix B

Publications

This research activity has led to several publications in international journals and conferences. These are summarized below.[1]

International Journals

1. **I.Zaza** A. Crisci, V. Grasso, P.Nesi, G. Pantaleo I.Paoli. "Predicting TV programme audience by using twitter based metrics", in *MULTIMEDIA TOOLS AND APPLICATIONS*, pp.1-30, 2017. [DOI:10.1007/s11042-017-4880-x]
2. **I.Zaza**, A. Crisci, V. Grasso, P.Nesi, G. Pantaleo, B.Gozzini. "Italian codified hashtags for weather warning on Twitter. Who is really using them?" in *ADVANCES IN SCIENCE AND RESEARCH*,pp.0-0., 2017. [ISSN:1992-0636]
3. **I.Zaza**, A. Crisci, V. Grasso, P.Nesi, G. Pantaleo, F. Zabini. "Weather events identification in social media streams: tools to detect their evidence in Twitter" in *Peerj*, 5th June

International Conferences and Workshops

1. **I.Zaza**, P.Bellini, P. Nesi. " RAISO: RAilway Infrastructures and Signaling Ontology for Configuration Management (or Validation and Verification", in *Proc. of Tenth IEEE International Conference on Semantic Computing (IEEE ICSC2016)*, Laguna Hills (USA), 2016.

[1]The author's bibliometric indices are the following: H-index = 1, total number of citations = 2 (source: Google Scholar on Month 09, 2017).

2. **I.Zaza**, P. Nesi, G. Pantaleo. "Real time Twitter Vigilance: Models and Tools for Monitoring City Events", in *Proc. of I-CiTies 2016 - 2nd CINI Annual Conference on ICT for Smart Cities & Communities*, Bari (Italy), 2016.
3. **I.Zaza**, P. Nesi, G. Pantaleo, D. Cenni. "Twitter Vigilance: a Multi-User platform for Cross-Domain Twitter Data Analytics, NLP and Sentiment Analysis", in *Proc. of 2017 IEEE 3rd International Smart World Congress (SmartWorld 2017)*, Laguna Hills (USA), 2017.

National Conferences

1. **I.Zaza**, P. Nesi, I.Paoli, G. Pantaleo, D. Cenni. "Twitter Vigilance: Modelli e Strumenti per l'Analisi e lo Studio di Dati Social Media ed il Monitoraggio in Real Time", in *Garr Conference 2016*, Firenze (Firenze), Italy, 2016.

Technical Reports

1. **I.Zaza** "Report of a comparative analysis of the Interlocking Systems"
2. **I.Zaza** "RAISSS project: Methods, notation and tools for modeling Command and Control Systems"
3. **I.Zaza**, C. Badii, A. Difino, M. Paolucci, "Moduli di connessione con SII per mobile, web e applicazioni fisse", Sii-Mobility-DE3-18a-ModConSIIMobWebApp

Bibliography

[1] ERTMS. [Online]. Available: http://www.ertms.net/

[2] ERTMS. [Online]. Available: http://www.railwaysignalling.eu/wp-content/uploads/2016/09/ERTMS_ETCS_signalling_system_revF.pdf

[3] ERTMS. [Online]. Available: http://www.era.europa.eu/Document-Register/Documents/SUBSET-091%20v330.pdf

[4] Railway Technical Web Pages. [Online]. Available: http://www.railway-technical.com/atpsurvey.shtml

[5] Skos simple knowledge organization system.

[6] "Formalising railway interlocking systems," 1998.

[7] "A hadoop based platform for natural language processing of web pages and documents," *J. Vis. Lang. Comput.*, vol. 31, no. PB, pp. 130–138, Dec. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jvlc.2015.10.017

[8] (2017) Apache hadoop. [Online]. Available: http://hadoop.apache.org

[9] (2017) Apache hbase. [Online]. Available: http://hbase.apache.org

[10] (2017) Brandwatch. [Online]. Available: https://www.brandwatch.com/

[11] (2017) Followerwonk. [Online]. Available: https://moz.com/followerwonk/

[12] (2017) Internet live stats @ONLINE. [Online]. Available: http://www.internetlivestats.com/

[13] (2017) Internet live stats @ONLINE. [Online]. Available: @ONLINE{ils:2017:Online,title={ApacheZookeeper{@ONLINE}}, year={2017},url={http://www.internetlivestats.com/}}

[14] (2017) Keyhole. [Online]. Available: keyhole.co

[15] (2017) Lyli indxer. [Online]. Available: https://github.com/NGDATA/hbase-indexer

[16] (2017) Mugnone 2016. [Online]. Available: http://protezionecivile.comune.fi.it/?p=7822

[17] (2017) Sas. [Online]. Available: https://www.sas.com

[18] (2017) Tweetracker. [Online]. Available: http://tweetracker.fulton.asu.edu

[19] (2017) Tweetreach. [Online]. Available: http://tweetreach.com

[20] (2017) Tweetris. [Online]. Available: http://tweetris.knoesis.org

[21] (2017) Twitter vigilance. [Online]. Available: https://www.disit.org/tv

[22] (2017) Twitter vigilance dashboard. [Online]. Available: http://www.km4city.org/dashboard/tv.html

[23] A. Abbasi, T. Fu, D. Zeng, and D. Adjeroh, "Crawling credible online medical sentiments for social intelligence," in *Social Computing (SocialCom), 2013 International Conference on.* IEEE, 2013, pp. 254–263.

[24] A. Abbasi, A. Hassan, and M. Dhar, "Benchmarking twitter sentiment analysis tools." in *LREC*, vol. 14, 2014, pp. 26–31.

[25] S. Agerholm and P. G. Larsen, "A lightweight approach to formal methods," in *International Workshop on Current Trends in Applied Formal Methods.* Springer, 1998, pp. 168–183.

[26] F. Baader, *The description logic handbook: Theory, implementation and applications.* Cambridge university press, 2003.

[27] M. Banci and A. Fantechi, "Some experiences on formal specification of railway interlocking systems using statecharts."

[28] ——, "Geographical versus functional modelling by statecharts of interlocking systems," *Electronic Notes in Theoretical Computer Science*, vol. 133, pp. 3–19, 2005.

[29] A. Barr, E. Feigenbaum, and P. Cohen, *The Handbook of artificial intelligence*, ser. The Handbook of Artificial Intelligence. HeurisTech Press, 1982, no. v. 3. [Online]. Available: https://books.google.it/books?id=poVQAAAAMAAJ

[30] B. Batrinca and P. C. Treleaven, "Social media analytics: a survey of techniques, tools and platforms," *AI & SOCIETY*, vol. 30, no. 1, pp. 89–116, 2015.

[31] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier, "Meteor: A successful application of b in a large project," in *International Symposium on Formal Methods.* Springer, 1999, pp. 369–387.

[32] P. Bellini, P. Nesi, and I. Zaza, "Raiso: Railway infrastructures and signaling ontology for configuration management, verification and validation," in *Semantic Computing (ICSC), 2016 IEEE Tenth International Conference on.* IEEE, 2016, pp. 350–353.

[33] P. Bellini, M. Benigni, R. Billero, P. Nesi, and N. Rauch, "Km4city ontology building vs data harvesting and cleaning for smart-city services," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 827–839, 2014.

[34] P. Bellini, R. Mattolini, and P. Nesi, "Temporal logics for real-time system specification," *ACM Computing Surveys (CSUR)*, vol. 32, no. 1, pp. 12–42, 2000.

[35] T. Berners-Lee and M. Fischetti, *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Paw Prints, 2008. [Online]. Available: https://books.google.it/books?id=Unp4PwAACAAJ

[36] L. Y. Bernhard Lorenz, Hans Jurgen Ohlbach. (31 August 2005) Ontology of transportation networks. [Online]. Available: http://rewerse.net/deliverables/m18/a1-d4.pdf

[37] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of computational science*, vol. 2, no. 1, pp. 1–8, 2011.

[38] M. Bosschaart, "Lean engineering design of rail interlocking systems with railml," 2013.

[39] M. Bosschaart, E. Quaglietta, B. Janssen, and R. M. Goverde, "Efficient formalization of railway interlocking data in railml," *Information Systems*, vol. 49, pp. 126–141, 2015.

[40] 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.

[41] A. Bruns and S. Stieglitz, "Twitter data: what do they represent?" *it-Information Technology*, vol. 56, no. 5, pp. 240–245, 2014.

[42] E. Cambria and A. Hussain, *Sentic computing: a common-sense-based framework for concept-level sentiment analysis*. Springer, 2015, vol. 1.

[43] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[44] P. Chikersal, S. Poria, E. Cambria, A. Gelbukh, and C. E. Siong, "Modelling public sentiment in twitter: using linguistic patterns to enhance supervised learning," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2015, pp. 49–65.

[45] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM*

Transactions on Programming Languages and Systems (TOPLAS), vol. 8, no. 2, pp. 244–263, 1986.

[46] E. M. Clarke, W. Klieber, M. Nováček, and P. Zuliani, "Model checking and the state explosion problem," in *Tools for Practical Software Verification*. Springer, 2012, pp. 1–30.

[47] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog *et al.*, "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.

[48] A. Crisci, V. Grasso, P. Nesi, G. Pantaleo, I. Paoli, and I. Zaza, "Predicting tv programme audience by using twitter based metrics," *Multimedia Tools and Applications*, pp. 1–30, 2017.

[49] C. A. Davis, G. L. Ciampaglia, L. M. Aiello, K. Chung, M. D. Conover, E. Ferrara, A. Flammini, G. C. Fox, X. Gao, B. Gonçalves *et al.*, "Osome: the iuni observatory on social media," *PeerJ Computer Science*, vol. 2, p. e87, 2016.

[50] B. P. Douglass, "Real-time uml." Springer.

[51] C. Eisner, "Using symbolic model checking to verify the railway stations of hoorn-kersenboogerd and heerhugowaard." Springer, 1999.

[52] L.-H. Eriksson and K. Johansson, "Using formal methods for quality assurance of interlocking systems," *WIT Transactions on The Built Environment*, vol. 37, 1998.

[53] A. Esuli and F. Sebastiani, "Sentiwordnet: a high-coverage lexical resource for opinion mining."

[54] A. Fantechi, "Distributing the challenge of model checking interlocking control tables," in *Proceedings of the 5th International Conference on Leveraging Applications of Formal Methods, Verification and Validation: Applications and Case Studies - Volume Part II*, ser. ISoLA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 276–289. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34032-1_26

[55] W. Fokkink, "Safety criteria for hoorn-kersenboogerd railway station," in *Utrecht University*, 1995.

[56] E. C. for Electrotechnical Standardization, "Railway applications communications, signalling & processing systems, software for railway control and protection systems," Tech. Rep., 2001.

[57] ——, "Communication, signalling and processing systems - safety related electronic systems for signalling," Tech. Rep., 2003.

[58] ——, "Railway applications applications - the specification and demonstration of reliability, availability, maintainability and safety (rams)," Tech. Rep., 2012.

[59] A. L. Gonzalez, D. Izidoro, R. Willrich, and C. A. Santos, "Ourmap: Representing crowdsourced annotations on geospatial coordinates as linked open data." in *CRIWG*. Springer, 2013, pp. 77–93.

[60] M. F. Goodchild and J. A. Glennon, "Crowdsourcing geographic information for disaster response: a research frontier," *International Journal of Digital Earth*, vol. 3, no. 3, pp. 231–241, 2010.

[61] V. Grasso, A. Crisci, M. Morabito, P. Nesi, and G. Pantaleo, "Public crowdsensing of heat waves by social media data," *Advances in Science and Research*, vol. 14, pp. 217–226, 2017. [Online]. Available: https://www.adv-sci-res.net/14/217/2017/

[62] V. Grasso, A. Crisci, M. Morabito, P. Nesi, G. Pantaleo, I. Zaza, and B. Gozzini, "Italian codified hashtags for weather warning on twitter–who is really using them?" *Advances in Science and Research*, vol. 14, p. 63, 2017.

[63] V. Grasso, I. Zaza, F. Zabini, G. Pantaleo, P. Nesi, and A. Crisci, "Weather events identification in social media streams: tools to detect their evidence in twitter," *PeerJ preprints*, vol. 4, p. e2241v1, 2016.

[64] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquis.*, vol. 5, no. 2, pp. 199–220, Jun. 1993. [Online]. Available: http://dx.doi.org/10.1006/knac.1993.1008

[65] M. Grüninger and M. S. Fox, "Methodology for the design and evaluation of ontologies," 1995.

[66] D. Harel and E. Gery, "Executable object modeling with statecharts," in *Proceedings of the 18th international conference on Software engineering*. IEEE Computer Society, 1996, pp. 246–257.

[67] A. E. Haxthausen, "An introduction to formal methods for the development of safety-critical applications," 2010.

[68] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular actor formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, ser. IJCAI'73. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235–245. [Online]. Available: http://dl.acm.org/citation.cfm?id=1624775.1624804

[69] G. Hilderink, A. Bakkers, and J. Broenink, "A distributed real-time java system based on csp," in *Object-Oriented Real-Time Distributed Computing, 2000.(ISORC 2000) Proceedings. Third IEEE International Symposium on*. IEEE, 2000, pp. 400–407.

[70] C. A. R. Hoare, "A model for communicating sequential process," 1980.

[71] T. Hoeve, "Model based testing of a plc based interlocking system," Master's thesis, University of Twente, 2012.

[72] K. Kanso, "Agda as a platform for the development of verified railway interlocking systems." Ph.D. dissertation, 2012.

[73] D. Kleindienst, R. Pfleger, and M. Schoch, "The business alignment of social media analytics." in *ECIS*, 2015.

[74] N. König, "The euro-interlocking project standards for interlocking systems in europe project," *Presentation for Polish Railways*, vol. 8, 2004.

[75] A. Lawrence, "Verification of railway interlockings in scade," 2011.

[76] T. Lecomte, T. Servat, G. Pouzancre *et al.*, "Formal methods in safety-critical railway systems."

[77] M. Lodemann, N. Luttenberger, and E. Schulz, "Semantic computing for railway infrastructure verification," in *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*. IEEE, 2013, pp. 371–376.

[78] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[79] N. R. Milton, *Knowledge technologies*. Polimetrica sas, 2008, vol. 3.

[80] M. Morley, *Safety Assurance in Interlocking Design*, ser. CST-129-96. University of Edinburgh, Department of Computer Science, 1996. [Online]. Available: http://books.google.it/books?id=dx3YGwAACAAJ

[81] A. Nash, D. Huerlimann, J. Schutte, and V. P. Krauss, "Railml a standard data interface for railroad applications," *WIT Transactions on The Built Environment*, vol. 74, 2004.

[82] N. F. Noy, D. L. McGuinness *et al.*, "Ontology development 101: A guide to creating your first ontology."

[83] M. Oussalah, F. Bhat, K. Challis, and T. Schnier, "A software architecture for twitter collection, search and geolocation services," *Knowledge-Based Systems*, vol. 37, pp. 105–120, 2013.

[84] C. Parsia. Stardog. [Online]. Available: http://www.stardog.com

[85] E. Pianta, L. Bentivogli, and C. Girardi, "Developing an aligned multilingual database," in *Proc. 1st Int'l Conference on Global WordNet*, 2002.

[86] R. Project. Rewerse project. [Online]. Available: http://rewerse.net

[87] A. Psyllidis, "Ontology-based data integration from heterogeneous urban systems: A knowledge representation framework for smart cities," in *CUPUM 2014: Proceedings of the 14th International Conference on Computers in Urban Planning and Urban Management, Cambrigde, USA, 7-10 July 2015*. MIT, 2015.

[88] E. research project. Intelligent integration of railway systems. [Online]. Available: http://www.iness.eu/

[89] C. Roussey, F. Pinet, M. A. Kang, and O. Corcho, "An introduction to ontologies and ontology engineering," in *Ontologies in Urban Development Projects*. Springer, 2011, pp. 9–38.

[90] U. Ruhi, "Social media analytics as a business intelligence practice: current landscape & future prospects," 2014.

[91] L. J. Sheela, "A review of sentiment analysis in twitter data using hadoop," *International Journal of Database Theory and Application*, vol. 9, no. 1, pp. 77–86, 2016.

[92] A. Sheth, A. Jadhav, P. Kapanipathi, C. Lu, H. Purohit, G. A. Smith, and W. Wang, "Twitris: A system for collective social intelligence," in *Encyclopedia of Social Network Analysis and Mining*. Springer, 2014, pp. 2240–2253.

[93] M. Skoric, N. Poor, P. Achananuparp, E.-P. Lim, and J. Jiang, "Tweets and votes: A study of the 2011 singapore general election," in *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE, 2012, pp. 2583–2591.

[94] A. N. Smith, E. Fischer, and C. Yongjian, "How does brand-related user-generated content differ across youtube, facebook, and twitter?" *Journal of Interactive Marketing*, vol. 26, no. 2, pp. 102–113, 2012.

[95] R. M. Smullyan, *First-order logic*. Courier Corporation, 1995.

[96] T. Soethout, "Applying path planning algorithms to train schedules," B.S. thesis, 2012.

[97] G. Staton, "Integrail project overview," *EGNOS/INTEGRAIL Demonstration, Guateng*, 2005.

[98] D. Tombs, N. Robinson, G. Nikandros *et al.*, "Signalling control table generation and verification," *CORE 2002: Cost Efficient Railways through Engineering*, p. 415, 2002.

[99] D. Tunkelang, "Faceted search," *Synthesis lectures on information concepts, retrieval, and services*, vol. 1, no. 1, pp. 1–80, 2009.

[100] P. Umiliacchi, U. Henning, G. Langer, and R. Shingler, "Standardized data interchange between railway systems: an integrated railway information system."

[101] 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.

[102] L. Zigterman, *A New Approach to Specification and Design of a Railway-interlocking: The "route" Approach. An Exercise in Top-down Design with Stepwise Refinement.* Nederlandse spoorwegen, 1984.