Contract No. H2020 –826172

SEMANTICS FOR PERFORMANT AND SCALABLE INTEROPERABILITY OF MULTIMODAL TRANSPORT

D4.1 - Analysis of the state-of-the-art and best practices in semantic automation for service integration

Due date of deliverable: 30/04/2019
Actual submission date: 20/05/2019

Leader/Responsible of this Deliverable: POLIMI

Reviewed: Y

Document status		
Revision	**Date**	**Description**
0.7	23/04/2019	First complete draft
0.9	30/4/2019	Revised version of complete draft
0.95	06/05/2019	Second revision of complete draft
0.99	08/05/2019	First Version
1.0	09/05/2019	Consolidated version for TMC approval
1.1	17/05/2019	Final version after TMC approval and Quality check

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/12/2018 Duration: 25 months

EXECUTIVE SUMMARY

This deliverable analyzes the state of the art that is relevant for the future developments of semantic-based technologies for the Shift2Rail Interoperability Framework, in particular for what concerns ontology creation and maintenance and the creation of mappings between data models. The deliverable first provides an assessment of the semantic technologies developed in previous Shift2Rail projects. Then, it analyzes the state of the art concerning various topics in semantic-based techniques, namely ontology engineering, automated ontology creation (through learning or extraction from existing models), data mapping techniques, and semantic-based service discovery. Finally, it concludes with a discussion pointing out the techniques and technologies that seem the most suitable to be the basis for the future developments of the semantic technologies related to the Interoperability Framework.

ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
API	Application Programming Interface
BF	Base Form
CSV	Comma-Separate Values
DCAT	Data Catalogue Vocabulary
DM	Direct Mapping
ETL	Extract, Transform, and Load
EU	European Union
FSM	Full Service Model
GA	Grant Agreement
H2020	Horizon 2020 framework programme
IF	Interoperability Framework
JSON	JavaScript Object Notation
JU	Shift2Rail Joint Undertaking
NOR	Non-Ontological Resources
NP	Noun Phrase
OAS	API Specification
OBDA	Ontology-based Data Access
ODP	Ontology Design Pattern
OWL	Web Ontology Language
PoS	Part of Speech
R2RML	RDB2RDF Mapping Language
RDB	Relational Data Base
RDF	Resource Description Framework
REST	Representational State Transfer
RML	RDF Mapping Language
RSP	Rail Service Provider

S2R	Shift2Rail
SPARQL	SPARQL Protocol and RDF Query Language
TAP	Telematic Applications for Passenger
TSI	Technical Specification for Interoperability
TSP	Transport Service Provider
XML	eXtensible Markup Language
WSMO	Web Service Modelling Ontology

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This deliverable discusses the state of the art in semantic technologies that are relevant for the development of the components of the Interoperability Framework (IF). The analysis carried out in this deliverable will be used to guide the developments of the semantic technologies carried out in Task 4.2 “Definition of a reference solution to automate the generation of ontologies, mappings and annotations” and in Task 4.3 “Analysis and selection of semantic tools and products and identification of missing features” of the SPRINT project.

The analysis starts (Section 2) from the technologies developed in previous Shift2Rail (S2R) projects, and in particular the IT2Rail [1], ST4RT [2] and GOF4R [3] projects.

Then, Section 3 analyzes various general-purpose semantic techniques that have been developed in the literature concerning ontology engineering, ontology learning, mappings between data models, and semantic-based discovery of services.

Finally, Section 4 summarizes the findings of the analysis of the previous sections, and highlights which are the most promising approaches for the future developments of the S2R semantic technologies.

2. ASSESSMENT OF THE OUTCOMES OF PAST SHIFT2RAIL PROJECTS

This section discusses the results of past S2R projects concerning, on the one hand, the semantic technologies they developed and, on the other hand, the mechanisms and policies they defined concerning the management of semantic artefacts. In particular, Section 2.1 discusses the semantic technology developed by the IT2Rail project [1]; Section 2.2 provides an assessment of the converter technology developed in the ST4RT project [2]; and Section 2.3 discusses the governance issues highlighted by the GOF4R project [3].

2.1 IT2RAIL

Advanced ICT solutions that can provide a truly customer-centric, one-stop-shop experience for multimodal travel across the Single European Transport Area must overcome the technical challenge of distributed computing: the ability to coordinate the execution of complex computational tasks that are inherently distributed on multiple heterogeneous systems, or “nodes”, of an open network with no central control. In this light, systems are interoperable if they are capable of participating in such distributed computing tasks.

Conventional approaches to the solution of this problem have concentrated in the past on altering artificially the essential features of the distributed computing landscape

- The adoption or regulation of common formats and protocols for inter-process communication aimed at removing heterogeneity.
- The local importation of remote data sets (data exchange) aimed at removing the distributed nature of data resources.
- The centralized governance of participant Actors in the scope of multimodal solutions aimed at controlling the openness of the network.

While designed to reduce the complexity of the technical challenge, these approaches reduce interoperability to controlling the movement of data sets across the network. However, they generate high costs in the adaptation of existing systems to common formats and protocols, in the administration and maintenance of these formats and protocols to keep them common in the face of changing requirements, and in forcing participants into a centrally synchronized roadmap for the deployment of solutions.

In contrast, the IT2Rail project has recognized one-stop-shopping for multimodal travel solutions as a natively distributed computing problem.

The “nodes” participating in a travel shopping, booking and ticketing process instance are independent “Travel Experts”, which control local resources embodying the “expertise” of a specific Transport Service Provider (TSP) Company about the travel solutions it provides – e.g. itineraries, modes of transport, prices and ticketing options. Shopping, Booking and Ticketing “orchestrators” developed in the project build a full solution that matches a specific Customer requirement by accessing and combining resources provided by Travel Experts over the network.

Definition:

The S2R IF encapsulates the “mechanics” of interoperability across the networked heterogeneous Travel Experts. It uses “semantic interoperability” principles and technology, described below, to achieve the following innovation objectives:

- insulate IT2Rail applications such as the Shopping and Booking orchestrators from the heterogeneity of TSP systems, providing an abstraction, the “web of transportation”, of the distributed data and computational resources available over the network;
- minimize or eliminate altogether the need for adaptation of TSP systems to become part of a network of services available to Customer experience applications;
- minimize the need for static exchange of data sets;
- minimize the need for centralized deployment roadmaps.

Semantic Interoperability refers to the ability of interacting computers to automate the interpretation of the data they process regardless of how this data is structured or exchanged. Knowledge about the domain problem, which is typically held by human analysts and programmers, is formalized in a set of logical statements, or “axioms”, written in a standard computer language available for machine processing. Human knowledge is thus transferred to machines and shared by them. Any particular representation of concepts and relationships in a specific data structure is associated, through a process of annotation, with its interpretation in terms of the domain problem. Machines can therefore discover and leverage equivalence relationships between different data formats with common meaning, and automate, therefore, the translation across these formats. Automated computer logical inference replaces therefore human programming of software to operate on different but equivalent data formats however they may be exchanged.

Implementation:

The Interoperability Framework is built on the principles of the ISO/EIC 10746 standard for Open Distributed Processing systems [4], using open source frameworks, allowing for multiple concurrent deployment options that can be tailored to specific operating environments. It exposes a set of specialized “Packaged Resolvers” – i.e. web services for use by IT2Rail applications to provide specific functions.

- **Location Identification** returns geographical coordinates of Locations that a Traveller requests by name.
- **Locations Resolver** returns a list of Stop Places within a requested radius from a point specified by its geographical coordinates. It is used during the Shopping process to identify transportation stops in the vicinity of Locations selected by Travelers from the list returned by Location Identification.
- **Network Statistics Provider** generates “meta routes” operated by TSPs. These “meta routes” are elements in the construction of meta-network used by the Shopping process.
- **Travel Expert Resolver** identifies Travel Expert and Booking Engine web services that can generate offers and bookings for specified “meta travel episodes” that satisfy a Traveller’s mobility request at the time of Shopping and Booking. It is used by the orchestrators to identify the subset of networked Travel Experts that participate in a coordinated distributed shopping and booking one-stop-shop instance.

- **Navitia Decoder** associates Stop Places and Transportation Services with the encodings used by the Navitia platform[1] for use by Trip Tracking in the identification of disruptions
- **Travel Expert** and **Booking Engine Brokers** mediate the interaction between the Shopping and Booking orchestration functions, respectively, and the Travel Expert or Booking Engine services provided by TSPs for the generation of offers and bookings that satisfy Traveller's mobility requests. Using semantic interoperability inferences, they perform the appropriate data transformations.

All "Packaged Resolvers" use a common underlying framework that handles the semantic interoperability mechanism described above and is controlled by inference rules and configuration information stored in the IF's **Asset Manager**.

The IF Assets Manager provides the tools that allow independent TSPs to participate in the "web of transportation" environment:

- The **Ontology Repository** stores the domain's knowledge represented as first order logic statements in the OWL language.
- The **Semantic Web Service registry** contains web service descriptors of the services exposed by participating TSPs – e.g. Travel Experts, Booking Engines. Descriptors are associated with semantically annotated data structures and inference rules that are used by the semantic interoperability mechanism to automate the conversion across different data structures.
- The **Triple Store** contains semantic graphs that describe resources such as Stop Places.

The Assets Manager supports workflows for versioning, approval and publication of shared resources such as the Ontology or Web Services descriptors.

The following external Travel Expert services have been annotated for use in the implementation:

- SNCF (mainline French Rail) PAO services
  - <endpoint>/it2r/sales/searchSolutions
- AMS (long distance Coach operators, Czech Republic) eshopcv services
  - <endpoint>/v1/Connection
  - <endpoint>/v1/ConnectionInfo
- Trenitalia (mainline Italian Rail) PICO Services
  - <endpoint>/Sale/SaleProcess/SolutionEngine/TravelSolution/search
  - <endpoint>/Sale/SaleProcess/SaleCoordinator/searchBase
- RENFE (mainline Spanish Rail), Indra Rail services
  - <endpoint>/Rail_TSP/NewTSP2/GetItineraries
  - <endpoint>/Rail_TSP/NewTSP2/Availability

[1] https://www.navitia.io/

- <endpoint>/Rail_TSP/NewTSP2/Trains
- KGOVV (Austrian Public Transport), HaCon services
  - <endpoint>/openapi/vao/restproxy/trip
- TMB (Madrid, Barcelona Public Transport) Indra Rail services
  - <endpoint>/HMI2_APP/service/otp/getRoute
- VBB (Belin / Brandenburg Public Transport), HaCon services
  - <endpoint>/restproxy/trip

The following external Booking Engine services have been annotated for use in the implementation:

- SNCF (main line Rail operator), PAO services
  - <endpoint>/it2r/sales/bookProposals
  - <endpoint>/it2r/sales/createSalesContract
  - <endpoint>/it2r/sales/cancelBooking
  - <endpoint>/it2r/sales/cancelTickets
- AMS (long distance coach services), eshopcv services
  - <endpoint>/v1/SeatBlock/
  - <endpoint>/v1/Ticket/
- Trenitalia (main line Rail operator), PICO Services
  - <endpoint>/Sale/SolutionEngine/CatalogReservation
  - <endpoint>/Sale/SaleProcess/OrderProcess
- RENFE (mainline Rail operator) Indra Rail services
  - <endpoint>/Rail_TSP/NewTSP2/LockInventory
  - <endpoint>/Rail_TSP/NewTSP2/IssueToken
  - <endpopint>/Rail_TSP/NewTSP2/BookingInfo
  - <endpoint>/Rail_TSP/NewTSP2/ReleaseInventory
- VBB (Public Transport Berlin/Brandenburg) HaCon services
  - <endpoint>/shopping/ShoppingMessages/VBB/purchaseRequest
  - <endpoint>/ shopping/ShoppingMessages/VBB/retrieveRequest

Achieved objectives:

Objective	Achievement
Mask interoperability “mechanics” to Applications	Seven heterogeneous systems interoperating with zero changes to data structures or communications protocols

Cut cost and time for TSPs to participate in providing multi-modal travel solutions without a coordinated "roadmap" for deployment	New independently developed shopping and booking providers added to interoperability scope in the course of the project
Operate with existing data structure specifications	Ability to work with a subset of NeTEx CEN/CENELEC data standards (Stop Places) demonstrated
Allow for multiple concurrent deployment options that can be tailored to specific operating environments	IFs deployed simultaneously on multiple web server containers. Automated semantic conversion libraries configured to be deployed simultaneously in brokers, at client or at server sides.

Table 1 - Achieved objectives of IT2Rail IF

2.2 ST4RT

Building on the IF technology developed in the IT2RAIL project, the ST4RT project has delivered a "Converter" software artifact enabling bi-directional semantic mapping of FSM[2] and TAP-TSI messages in a specific use case – i.e., booking of a Berth on a Trenitalia night train traveling from Roma Termini to Palermo Centrale stations.

Some specific features of the TAP-TSI specification[3] have led additionally to the extension of the semantic annotations model with respect to the original IT2Rail IF software, to new terms added to the IT2RAIL ontology, and to the need to access data items external to the content of the exchanged FSM and TAP-TSI messages, which has been achieved by adding semantic graph federation capabilities to the framework

As a consequence, the ST4RT project constitutes an extension of the Interoperability Framework in that:

- it adds the ability to handle FSM/TAP-TSI message specification conversion;
- it adds terms to the Ontology;
- it adds features to the annotation model used to associate semantics to object specifications;
- it adds the ability to operate on federated distributed semantic graphs.

The implementation of these new features has been tested in the following scenarios:

1. "Pure" conversions
    a. TAP-TSI ReservationRequest to FSM PreBookingRequest
    b. TAP-TSI ReservationReply to FSM PreBookingResponse
    c. FSM PreBookingRequest to TAP-TSI ReservationRequest
    d. FSM PreBookingResponse to TAP-TSI ReservationReply

[2] https://tsga.eu/fsm_login
[3] http://taf-jsg.info/?page_id=51

2. TAP-TSI ReservationRequest / Reply transaction to an FSM Simulator
3. FSM PreBookingRequest/Response transaction to a TAP-TSI Processor
4. FSM Offering process to IT2Rail Travel Expert Broker for shopping.

The extended IF with the added ST4RT features has been deployed and run in the same unchanged demonstration environment used for the initial IT2RAIL project.

2.2.1 FSM and TAP-TSI scenario differences

An important non-technical outcome of the project has been the identification of conceptual differences in the usage of TAP-TSI and FSM message design. These differences are *conceptual* in the sense that they are independent on *any* conversion mechanism that may be used and, *a fortiori*, on semantic conversion in particular. They impose restrictions on how and when FSM and TAP-TSI specifications can be used regardless of any conversion mechanism, but once this restricted scenario is identified it can be supported by semantic conversion. The conceptual differences in the design are illustrated in the following figure

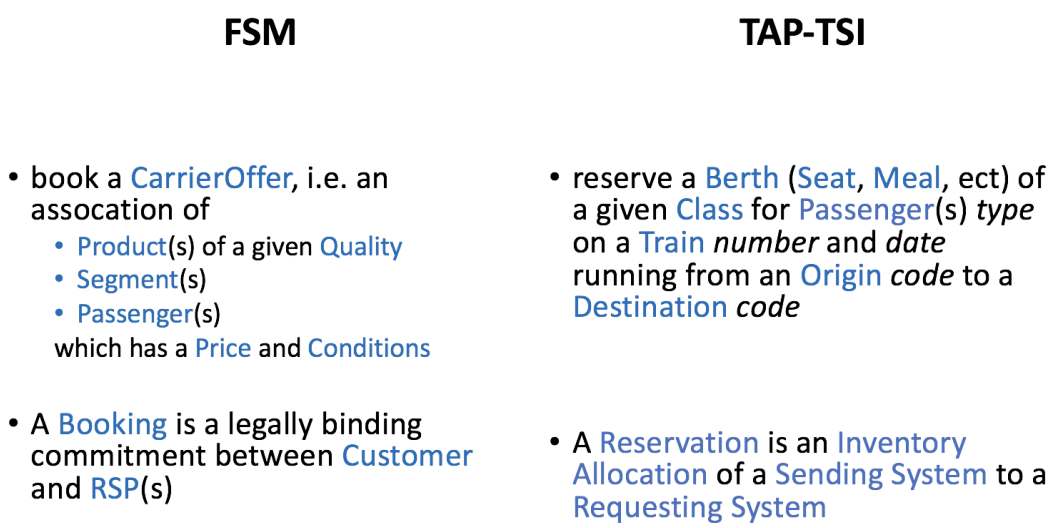

Figure 1 - FSM and TAP-TSI booking/reservation scenario

An examination of Figure 1 leads to the following considerations:

1. In the FSM scenario a CarrierOffer is a full description of the “item for sale” that a Customer accepts to purchase at the given conditions (for example for refunds): it specifies which Passengers will consume which Products on which Segments, and at what price. FSM Products are generic – e.g. “Week-end getaway” – and are not necessarily inventory-controlled. A booking is always created for a CarrierOffer, representing the commitment of the Rail Service Provider (RSP) to provide the specified products to the passengers on the specified segments, and the commitment of the Customer or an Agent operating in his/her behalf, to compensate the provider. The booking request structure is the same for all CarrierOffers because the details of what is being booked is in the CarrierOffer, not the booking request.

2. By contrast, different TAP-TSI data structures must be used for each different bookable item, and there can be at most one such structure in a ReservationRequest. Unlike FSM Products, TAP-TSI bookable items are therefore restricted to those pre-defined in the TAP-TSI specification – e.g. Berth, Seat, Meal – and only one type can be booked at a time, meaning that all Passenger types in the same request must be allocated a unit of the same bookable item type. A Reservation Reply represents an Inventory Allocation and therefore it cannot be used to describe a purchase except for inventory-controlled items.
3. In the TAP-TSI scenario Passenger type, Train, Origin and Destination are represented by *codes*, whereas in the FSM scenario they are full descriptions of the corresponding entities. For example, in a TAP-TSI Reservation Request "Passenger" indicates the *number* of passengers of a given *type* – e.g., 1 Adult – not the actual Passenger's information such as name, age, etc.
4. In the TAP-TSI scenario a Reservation Request / Reply establishes a financially relevant relationship between Requesting and Sending Systems, and therefore between the Companies that "own" these systems. By contrast, in the FSM Scenario the financially relevant relationship is established by a separate payment and settlement process operating on a Booking that has no equivalent in the TAP-TSI specification.

2.2.2 Mixed FSM / TAP-TSI scenario restrictions

The preceding considerations further restrict the applicability of a mixed FSM / TAP-TSI scenario regardless of how data structures are converted into one another, as follows:

1. An FSM Booking request cannot contain CarrierOffers supplied by different RSPs.
2. An FSM CarrierOffer submitted through conversion to a TAP-TSI processor can include at most one Product, and this Product must be an inventory-controlled TAP-TSI bookable item – e.g., a Berth, Seat. If more than one Passenger is in the CarrierOffer, then all Passengers consume a unit of the same Product on the same Segment.
3. Passenger details, except passenger type such as "Adult", are ignored by the TAP-TSI processor.
4. The CarrierOffer's Price and Conditions are ignored by the TAP-TSI processor.
5. An implicit assumption must be accepted that a ReservationReply generated by a TAP-TSI processor – i.e., an Inventory Allocation – is a sale recorded between the "owner" of the FSM requesting system and the "owner" of the TAP-TSI processor. Payment from the Customer must be handled by the owner of the FSM requesting system.

2.2.3 Mixed FSM / TAP-TSI scenario conceptual mappings

Within the restrictions described in the previous section, the following conceptual mappings can be established between the FSM and TAP-TSI data structures:

1. An FSM PreBookRequest maps to/from a TAP-TSI ReservationRequest.
2. An FSM PreBookResponse maps to/from a TAP-TSI ReservationReply.

3. An FSM CarrierOffer's Product maps to a TAP-TSI bookable item – e.g., a Berth – and is used to generate the appropriate request in the ReservationRequest or allocation in the ReservationReply.
4. The StopPlaceCodes of the origin and destination StopPlaces of an FSM Itinerary Segment map to/from TAP-TSI Origin and Destination StationCodeType.
5. An FSM Segment's Vehicle's vehicleId maps to/from TAP-TSI Train number.
6. An FSM Caller maps to/from TAP-TSI Requestor and Terminal Type identifying the sending and receiving systems.
7. The number of Passenger instances in an FSM request, assumed to be all Adults in the use case, maps to/from the integer number of Adult passengers in the TAP-TSI messages.

At the high level all the main elements described in Figure 1 for the FSM and TAP-TSI scenarios were identified, indicating that, under the restrictions listed in the previous sections, it is logically possible to operate a mixed FSM / TAP-TSI scenario using the IF.

2.2.4 Mixed FSM / TAP-TSI scenario ontology development

In order to automate the conversion, the common high-level conceptual mappings where further analysed with respect to the initial IT2RAIL ontology, and details of both data structures were used to create a new, extended, ST4RT ontology used in the annotation process of the ST4RT converter. The outcomes of this activity are documented in the D3.3 Mapping between standard and reference ontology[4], and D4.3 Mapping between standard and reference ontology [5] deliverables. Due to the particular design of the TAP-TSI specification, and particularly of its usage of *codes* to describe meaning and data stored in external code lists, this process was found to be extremely labor-intensive. It also necessitated the creation of semantic graphs representing the code lists which, while adding to the effort, it also demonstrated that code lists can indeed be captured by a distributed semantic graph, opening up the possibility of eliminating the need to copy the lists as data sets from one system to another. In fact, such code list graphs can be stored in the IF's triple store – i.e. the data layer – and shared through the Asset Manager. Since the FSM specification does not have provision for 'master data', and allows for the use of code lists, this capability can be used for a purely FSM scenario using master data managed by the Asset Manager.

2.3 GOF4R

The GOF4R objective was to define sustainable governance for the IF that will create the right conditions to introduce seamless mobility services and foster the development of multi-modal travel services.

An essential outcome of the project was the realization that the "governance problem" is strictly associated with the mechanisms used for interoperability.

[4] http://www.st4rt.eu/download.aspx?id=0054e8e6-dcb4-4607-bd2d-5a8be3129caf

[5] http://www.st4rt.eu/download.aspx?id=eaacec02-94d2-4450-8a4a-8650f0d46167

interoperability mechanism	Governance problem
• Adoption of common formats and protocols	• How do we accommodate variants while *keeping* them common? How do we ensure *synchronized* “adoption”?
• Local importation of pre-defined remote data sets	• How do I know who gets my data sets, how they use them? • How do I know where they come from? • How do we ensure they are correct and up-to-date? • How do we ensure everybody using the same data *calculate* the same results?

Table 2 - interoperability by conventional "data exchange"

interoperability mechanism	Governance problem
• Adoption of shared machine-interpretable semantics (ontology) that *abstracts* from formats	• How do we *evolve the ontology?* • How do we *map* across ontologies?
• *Link* data across the web	• How we *discover* and *establish* links across the web of data? • How do we provide what *tools* and *education*?

Table 3 - semantic interoperability

Comparison of the conventional “data exchange” and semantic interoperability illustrated by Table 2 and Table 3 above shows that the adoption of semantic interoperability eliminates control of actual data flows from the governance mission, changing its nature into one of evolution of the formal conceptualization of the domain knowledge (ontology), and the meaningful linking of distributed data. Both the latter can be at least in part automated with appropriate advanced tooling. The SPRINT project is designed as a step forward in this direction.

3. ANALYSIS OF THE STATE-OF-THE-ART OF SEMANTIC TECHNOLOGIES FOR INTEROPERABILITY

This section surveys the state of the art in various topics that are relevant for the development of the semantic technologies that will be developed during the SPRINT project. First, it studies the problem of manual ontology creation and maintenance (Section 3.1); then, it looks at techniques for the automated creation of ontologies starting from unstructured information (Section 3.2), and also from structured notations such as UML (Section 3.2.1); further, Section 3.4 surveys techniques for creating mappings between data models; finally, Section 3.5 analyses semantic-based techniques for service discovery.

3.1 Ontology Engineering Techniques and Tools

Ontologies started to be built in the early nineties using domain-specific methods, tools and techniques that were typically driven by the targeted domain. The emergence of ontology development methodologies during the nineties started to transform the art of building ontologies into an engineering activity, moving from ad-hoc and domain-oriented methodologies into general-purpose methodologies. Of these methodologies, METHONTOLOGY [5] and On-To-Knowledge [6] were considered to be the most complete ones for building single ontologies from scratch. To deal with the collaborative and distributed development of ontologies, the DILIGENT methodology was proposed [7].

In general, all these methodologies include only high-level guidelines for single ontology construction. Such guidelines normally (a) cover from the specification to the implementation, but only few of them provide shallow guides for reusing ontologies, and (b) are mainly targeted to researchers, thus, they are not described following a user-oriented approach. Detailed surveys and comparative studies of these methodologies are provided in [8].

3.1.1 The NeOn methodology

In the context of the NeOn EU project (FP6-027595), the NeOn Methodology [9] [10] for ontology building was created. The NeOn Methodology Framework is a scenario-based methodology that provides prescriptive methodological guidelines for the development of ontology networks. The aim of this methodology is to accelerate the construction of ontologies and ontology networks by reusing available knowledge resources (ontologies, non-ontological resources and ontology design patterns).

Knowledge resource reuse is becoming a widespread approach in the ontology engineering field because it can speed up the ontology development process. In this context, the NeOn Methodology specifies some guidelines for reusing different types of knowledge-aware resources. These guidelines mainly cover the following activities: 1) search repositories and registries for candidate resources that could satisfy the needs of the ontology network under development; 2) assess whether the set of candidate knowledge-aware resources are useful for building the ontology network; 3) select the best candidate resources for developing the ontology network on the basis of a set of criteria; and, 4) integrate the selected resources into the ontology network under construction.

The key assets of the NeOn Methodology Framework are:

- A set of nine scenarios for building ontologies and ontology networks, emphasizing the reuse of ontological and non-ontological resources, the reengineering and merging.
    - Scenario 1: From specification to implementation. The ontology network is developed from scratch (without reusing existing resources). Developers should specify ontology requirements. After that, it is advised to carry out a search for potential resources to be reused. Then, the scheduling activity must be performed, and developers should follow the planning.
    - Scenario 2: Reusing and re-engineering non-ontological resources (NORs). Developers should carry out the NOR reuse process for deciding, according to the ontology requirements, which NORs can be reused to build the ontology network. Then, the selected NORs should be re-engineered into ontologies. See Section 3.2 for an instantiation of this scenario in the context of the so-called Ontology Learning.
    - Scenario 3: Reusing ontological resources. Developers use ontological resources (ontologies as a whole, ontology modules, and/or ontology statements) to build ontology networks.
    - Scenario 4: Reusing and re-engineering ontological resources. Ontology developers reuse and re-engineer ontological resources.
    - Scenario 5: Reusing and merging ontological resources. This scenario arises when several ontological resources in the same domain are selected for reuse, and developers wish to create a new ontological resource with the selected resources.
    - Scenario 6: Reusing, merging and re-engineering ontological resources. Ontology developers reuse, merge, and re-engineer ontological resources. This scenario is similar to Scenario 5, but here developers decide to re-engineer the set of merged resources.
    - Scenario 7: Reusing ontology design patterns (ODPs). Ontology developers access repositories to reuse ODPs.
    - Scenario 8: Restructuring ontological resources. Ontology developers restructure (e.g., modularize, prune, extend, and/or specialize) ontological resources to be integrated in the ontology network.
    - Scenario 9: Localizing ontological resources. Ontology developers adapt an ontology to other languages and culture communities, thus obtaining a multilingual ontology.
- The NeOn Glossary of Processes and Activities, which identifies and defines the processes and activities carried out when ontology networks are collaboratively built by teams.
- Methodological guidelines for different processes and activities of the ontology network development process, such as the reuse and reengineering of ontological and non-ontological resources, the ontology requirements specification, the ontology localization, the scheduling, among others. All processes and activities are described with (a) a guiding card, (b) a workflow, and (c) examples.

3.1.2 Collaborative ontology engineering with OnToology

The previous section has focused on a comprehensive ontology engineering methodology such as the NeOn methodology, which is the one recommended for the ontology development activities in the context of the IF.

In this section we focus on how some of the scenarios envisaged by the NeOn methodology are supported by tools, especially in the context of situations where a collaborative ontology engineering process is required (e.g., when multiple parties are contributing to the development of ontologies and want to make use of well-known software development practices in this context). Now that ontologies have been adopted in many software projects, it is clear that the ontology development phases, identified by the aforementioned methodologies, have to be included into the common software engineering practices used by software developers. Related work has already been done in the ontology engineering community to adapt ontology development to agile software development methodologies [11], and some initial work has been done on allowing collaborative ontology development throughout the use of common-practice software engineering tools (e.g., VoCol [12], Moki [13]).

In [14], OnToology is described as an open source web application that detects the changes made over a git repository and triggers a series of activities for supporting the evaluation and publication stages of the ontology development process. OnToology integrates a suite of existing tools, and uses the files used to serialize ontologies (e.g., OWL files) as the main resource to work with. It is currently integrated with GitHub and integrations with other source code repository technologies are planned as part of its development roadmap.

The steps to be used by a set of users working with OnToology for the development of their ontologies are presented in Figure 2. As it can be seen, the process is launched whenever a change in an ontology of a GitHub repository that is being watched by OnToology is pushed into GitHub. As a result of this, OnToology forks that repository and starts making use of a set of different tools related to the ontology development process (namely, Widoco for ontology documentation, AR2Dtool for diagram generation, Oops! for ontology evaluation, and vocabLite, if configured, for the generation of an ontology catalogue with the updated information). After all these tools/services are used, the results of their invocation are included in the corresponding forked repository and a pull request is made to the original GitHub repository, which can be accepted or not by the ontology developers in charge of the original repository.

Users can also force the invocation of all the related services at any point in time, if they consider this necessary for whatever reason or want some changes in the configuration files for the invocation of any of these services to be done. Furthermore, OnToology generates a bundle that can be used to publish the ontology and its documentation in a Web server according to Linked Data principles, therefore enabling content negotiation for the ontology file and its corresponding terms.

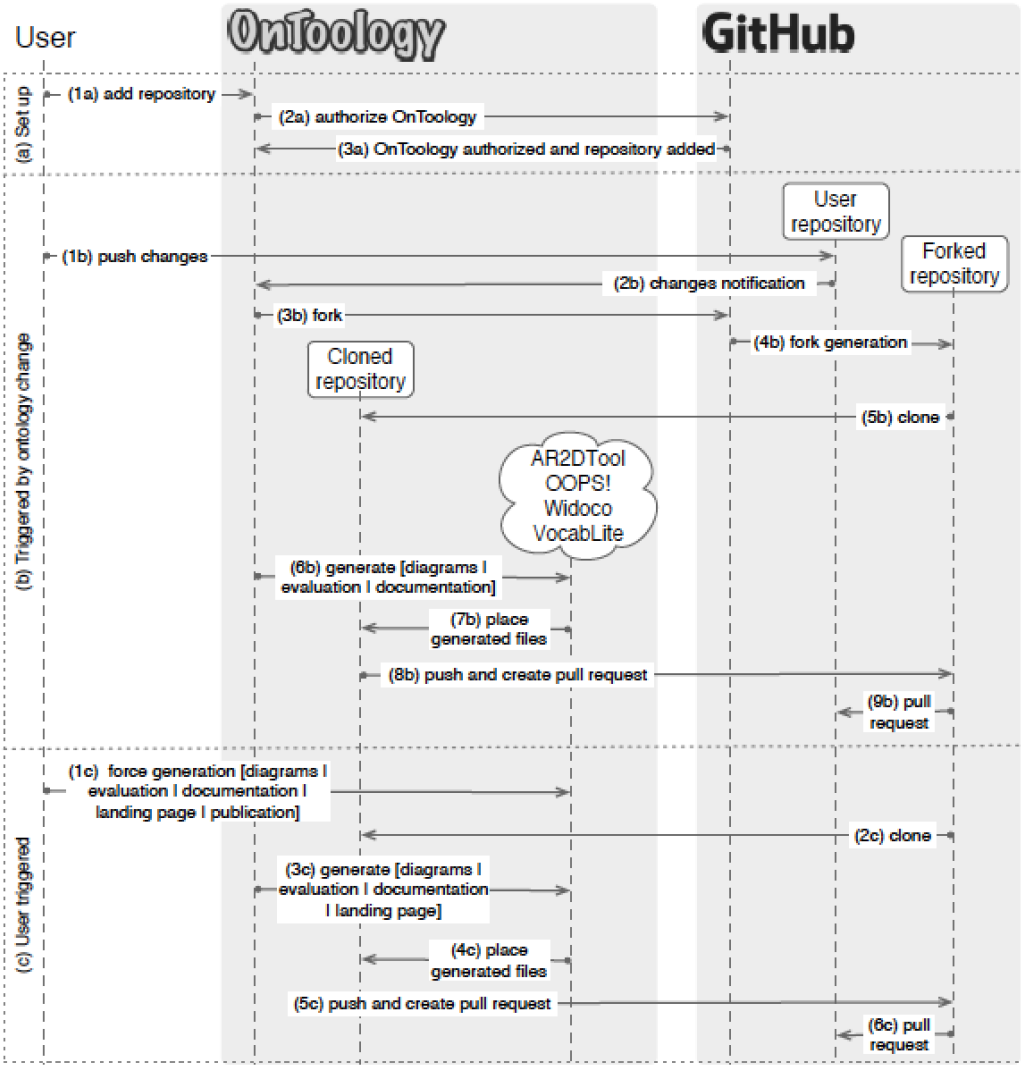

Figure 2. Steps followed by OnToology (taken from **[14]**).

3.1.3 Collaborative ontology engineering with other tools

In recent years, different systems have been developed to support teams in the distributed development of ontologies. One of the best-known tools is the WebProtégé editor [15]. Besides the ontology edition functionalities, WebProtégé provides a discussion board and functions for annotating ontology terms. Once an ontology is generated, developers may resort to their local installation of Protégé to produce human-readable documentation and diagrams using a variety of plug-ins.

Another approach is Moki [13], a collaborative tool for modeling ontologies based on MediaWiki. Moki provides either a light-weight view or a full source-code view of the ontology. It also integrates evaluation functionalities like model checklist and quality indicators. However, neither WebProtégé nor Moki integrate features for the online publication of the ontology.

Neologism [16] was also designed to provide support for the online development process of ontologies, as a Drupal-based vocabulary editor and publishing system. Neologism provided an automatic diagram creation that showed the classes and properties of a vocabulary. However, this system is no longer maintained.

A more recent effort is VoCol [12], designed as a tool to help collaborative vocabulary development, inspired by agile software and content development methodologies, and using Git repositories to maintain the vocabulary-related files. VoCol provides support for project management, quality assurance, documentation and visualization components. Both Neologism and VoCol provide a complete encapsulated framework to publish ontologies and their documentation, relying on the user to deploy it.

VocBench [17] is an open source web application for editing thesauri complying with the SKOS and SKOS-XL standards. VocBench allows for collaborative management of the overall editorial workflow, by introducing different roles with specific competencies, and provides features for content validation and publication of vocabularies. Furthermore, it provides a full history of changes and a SPARQL query service. However, VocBench does not provide any documentation or evaluation functionalities, and focuses only on SKOS models.

3.2 ONTOLOGY LEARNING

The techniques presented in Section 3.1 focused on the manual building of ontologies. Another – complementary, and not necessarily alternative – possibility for ontology creation is to automatically learn the elements of an ontology, which could then be manually refined as in Scenario 2 of Section 3.1.1. The steps of the ontology learning process (*ontology learning layer cake* [18]) include extracting terms, their synonyms, acquiring concepts and existing relationships among them in the text and, finally, inducing domain-related axioms and rules. In addition, a pre-processing task at the beginning of the process and an evaluation of the results at the end are necessary for enhancing the effectiveness of the learning. So far, several approaches have been proposed to tackle different stages of ontology learning which are summarized in the next sections.

3.2.1 State of the art

Up to now, various methodologies have been suggested so that the process of ontology learning from texts can be automatized, which include techniques in the fields of machine learning, text mining, knowledge representation and reasoning, information retrieval and natural language processing.

These ontology learning techniques are classified into three main categories: linguistics, statistical and logical [19] and are applied to different stages of ontology learning task. Figure 3 illustrates the main stages and different layers of ontology learning.

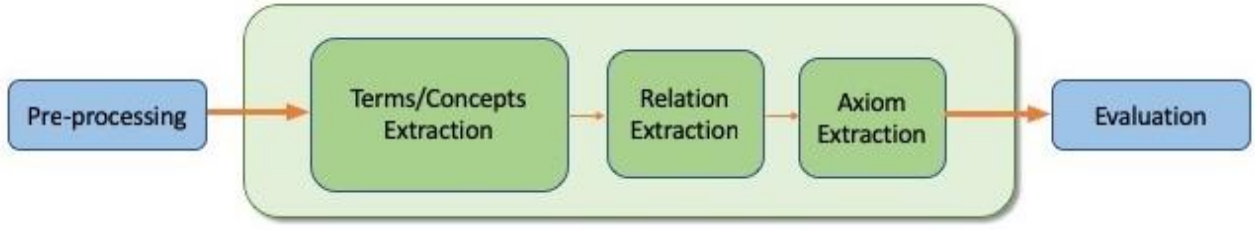

Figure 3. Ontology Learning Layer Cake

3.2.2 Pre-processing

In the pre-processing step of ontology learning the unstructured corpus should be ready for later tasks, by means of Natural Language Processing which provides linguistic tools. The most prominent techniques that are applied to the text in this stage are part of speech (PoS) tagging, parsing and lemmatization.

Part of speech tagging:

PoS tagging, grammatical tagging or word-category disambiguation is a linguistic technique for annotating words in the corpus with their corresponding part of speech, according to both its definition and its context. PoS algorithms fall into two categories, stochastic and rule based. A classical and widely used tool is Brill Tagger [20] which is a rule-based tagger capable of automatic learning of the rules. It is very simple and does not need large tables of statistics to capture contextual information. First, the tokenized words in the text are tagged according to their most probable part of speech in the dictionary and non-contextual features. Then, using iterative contextual rules and the errors from the previous stage, tags are transformed to reduce the amount of tagging errors. However, due to its inherent local non-deterministic behavior which reiterates the tags, it could be very slow in training and tagging stages (in the worst-case scenario the original implementation requires $R \cdot K \cdot n$ steps for tagging a sequence of n words, demanding K words and using R rules) [21, 22]. Tree Tagger is a stochastic PoS tagger method in which transition probabilities are estimated using a decision tree. This method of tagging achieves 96.36% accuracy. Nevertheless, since most of the words in English are unambiguous and uncertain parts of speech of words are rarely used, assigning just the most probable tag without further transitions give an accuracy close to 90% [23].

Parsing:

Parsing – or syntactic analysis – is the process of analyzing the grammatical structure of the words of sentences and building the corresponding parse tree. Principar [24] is a principle-based parser for English language which is implemented in C++ according to the algorithm proposed in [25]. It applies a set of extraction and conversion rules to a lexicon with over more than 90000 entries. In addition, Link Grammar Parser is an open source tool written in C which develops algorithms for efficiently parsing in English by providing a formal grammatical system to encode the English grammar. Among statistical parsing systems, Stanford Parser utilizes an unlexicalized probabilistic context-free grammar (PCFG) [26] to parse the text.

Lemmatization:

Lemmatization is a linguistic pre-processing method for identification of the words' lemma (dictionary form) in the text from their inflected variants and bring them back to their base form [27]. For example, the words swims, *'swimming'*, *'swam'* etc. should be converted to the dictionary form *'swim'*. Unlike stemming, lemmatization is dependent on the context surrounding the word and its part of speech, while a stemmer transforms words without considering the context; for example, the result of stemming for the word *'meeting'* is always *'meet'*, but a lemmatization algorithm would discriminate if it is a noun or verb. The consequence of this difference is that applying stemming would increase the recall, while the precision increases with lemmatization techniques. Both methods would reduce the dimensionality of the data [19].

According to [27], the main problems in data-driven lemmatization algorithms are the generalization capability in case of unseen words, and disambiguation. Several context-sensitive lemmatization algorithms utilize a hand-built morphological analyzer to rehash ambiguous words in languages [28], [29], [30], while some try to learn the lemmas (and tags) using the context to process unseen data, without an existing analyzer [31], [32]. However, there is no accurate measurement about to what extent the context would perform on unseen words. It is not only dependent on the amount of training data presented in a language, but also on the morphological regularity and characteristics of a language. Lematus [27] is a lemmatization system based on the neural machine translation framework [33] which uses an encoding and decoding model to learn context-sensitive lemmatization. Still, the actual relationships between the ambiguity level in a language, productivity[6], and their correlation with lemmatization accuracy is unclear. One of the linguistic analysis tools is Stanford CoreNLP API[7], which is written in Java and performs pre-processing tasks including lemmatization. This tool has been used in [34].

3.2.3 Ontology Learning Steps

Terms/Concepts Extraction:

The following are linguistic-based methods for domain-specific terms extraction:

Syntactic analysis is a NLP-based method for extracting the set of terms and concepts in the text. It utilizes the syntactic structural properties of compound words in the text and the head-modifier principle, which suggests how a major class of compounds are formed. For example, the structure of the noun phrase could be used for extracting possible hierarchical terminologies such as hyponymy and meronymy [19]. The concept of head and modifier exists in the grammar of many languages, which suggests that in the syntactic construction of phrases, one of the units (or chunks) is called the head – or core – part of the phrase, and other words modify it; also, the core part is associated with the semantics of the phrase [35]. Since the grammatical structure of the language is used in this method, it can be applied to structured texts and operates on other languages too. However, before applying syntactic analysis, the text should be tagged and parsed, so preprocessing techniques play an important role.

Subcategorization frame is a linguistic-based method for term/concept extraction [19], in which a set of rules are defined for generating and identifying different syntactic structures out of the base from (BF) of a given lexeme. This method is useful in case of a regular pattern and can be applied to generate constituents linked to the base form. For example, in English many verbs take a NP (Noun Phrase) as the subject (specifier) and another NP as direct object (complement). This grammatical structure can be defined by some deterministic rules (frames). In particular, a frame defines the number of words of a specific form a lexeme takes as its neighbors in a sentence. For example, the verb ‘to enjoy’ has two neighbors in the sentence ‘Bob enjoys reading’. Due to the inherent concept of subcategorization frame, it cannot be used in case of irregular behavior.

[6] https://en.wikipedia.org/wiki/Productivity_(linguistics)

[7] http://stanfordnlp.github.io/CoreNLP/

ASIUM [36] (Acquisition of SemantIc knowledge Using Machine learning method) is a cooperative[8] machine learning-based system for knowledge acquisition in technical texts and restricted domains and for learning subcategorization frames of verbs and ontologies; it also provides automatic concept splitting using syntactic parsing disambiguation without manual annotations. The evaluation of learnt concepts is provided during the training step, but validation and adjustment are needed specifically if the texts and parsing method are noisy.

Seed words are domain-specific terms which can be used as a base enhancement for other methodologies in order to extract terminologies. These initial keywords ensure that the words that are most semantically similar to the seeds are detected as the target concepts [19] [21] [22] [33] [37].

The following methods use statistical information and probabilistic methods for extracting terms, without considering their semantics.

C-Value/NC-Value is a domain-independent method for automated multi-word extraction in technical texts. It incorporates linguistic and statistical approaches and scores combinations of words according to the probability that they have a valid concept in the text. The scoring function is a mixture of two values: the C value and the NC value [38]. The C value tries to extract nested terms and produces a ranked list of multi-word candidates based on their termhood possibility. Prior to that it applies PoS tagging and creates a stop-list to filter candidates based on their types (tags), to improve precision and recall. Its ranking function considers the following figures:

1. the number of occurrences of the term
2. the frequency of appearance as part of longer candidate term
3. how many times the longer candidate has appeared
4. the length of the candidate string (words).

The NC value re-ranks the list by adding context information, which is a weighted sum of the words that have appeared with the candidate and can additionally reflect the real importance of a term in the domain, in the sense that neighbor words often reveal information about the term. For example, the termhood and importance of a NP appearing after the verb ‘call’ is higher. This way the algorithm benefits from this information, too. As a result, the concentration of real terms would increase compared to the first list. Then, the resulting ranked list can be scanned by experts to check the validity of the terms up to a certain point. Although the system is fully automated and independent from other sources such as external dictionaries, its outcome should still be validated by domain experts at the final stage.

The main idea of **Contrastive Analysis (Domain Relevance and Domain Consensus)** is to extract domain-related terms by filtering out the irrelevant ones [19]. OntoLearn [39] is an ontology engineering platform which uses contrastive analysis in the concept extraction stage. After retrieving a list of syntactically promising candidate terms (possibly multi-words), it measures two entropy-related values, namely domain relevance and domain consensus, using different domains. Domain relevance measures the specificity of the term with respect to the target domain, and more specifically the ratio of information given by terms in the target domain to the sum of the information it gives in all the domains. On the other hand, domain consensus suggests that if a word is a term used in a domain, it should appear in

[8] Asium is not fully automated, but provides interactive features during the learning task.

several documents of that domain. OntoLearn has been experimented in two European projects (FETISH and HARMONISE[9]) as the basic semantic interoperability system [40].

Co-occurrence analysis is a method to extract terms and relations among them by locating terms that co-occur in different forms such as phrases and relations. The strength of similarity among two terms or relations are quantified by different metrics, including Mutual Information, Chi-Square, Cosine and Dice Similarity [41].

Latent Semantic Analysis (LSA) is a statistical method for finding concepts and relations. The main idea in LSA is that the words occurring together have more similar meanings [19]. Unlike co-occurrence analysis, it does not consider just pairwise co-occurrence of the terms; rather, it keeps a detailed pattern of which words have occurred in which document in the corpora. Firstly, it builds a matrix that represents the frequency of each term in each document. Then, Singular Value Decomposition (SVD) is applied to the term-document matrix in order to reduce the dimensionality without losing the similarity structure. In other words, the meaning of each term is a reflection of the meaning of the document containing the term and the meaning of the document is an average of the terms in the document. However, LSA does not benefit from the order of the words in the sentence, or from morphology [42].

Clustering is an unsupervised learning method which groups similar objects together. Among clustering techniques, **Contextual Concept Discovery (CCD)** [43] is a hierarchical clustering algorithm based on incremental segmentation of the data by employing k-means several times to extract ontological concepts. The algorithm clusters the candidate terms into three groups: the advisable group, which contain the set of terms validated by domain experts; the improper cluster, which contains terms with more than one concept; and the unknown group, which are terms without any semantic relation and which are not validated by domain experts. The algorithm performance was evaluated on HTML documents in the tourism domain.

Relation Extraction:

In addition to extracting concepts, the relation among the concepts should be extracted too. These relations may be taxonomic or non-taxonomic. In another categorization, the techniques for extracting concepts' relations are either statistical-based or linguistic-based. The following are the linguistic-based methods for relation extraction.

Dependency Analysis extracts relations according to the existing dependencies among the words in the sentence. First the sentences are parsed and tagged to obtain a syntax tree. For each tree a dependency graph is created, in which each morphologically simplified word is a node and is connected to the governor (syntactic head) of its closest neighbor phrase. The governor is the word in a phrase whose part of speech determines the syntactic category of the whole phrase. Finally, the shortest path between two entities in the tree, following the dependency relations, is their semantic relationship [44].

The **lexico-Syntactic Pattern** method looks for dependency paths in the dependency tree that represents the syntactic relations between words. It makes tuples of the words and the relations, where the relations are specific links between two words or phrases in the tree. As an example, [45] defined a lexico-syntactic patterns space such that it includes the links (the shortest paths) equal or less than four between any two nouns in a dependency tree.

[9] ITS-13015 (FETISH) and ITS-29329 (HARMONISE) [17].

The followings are the statistical methods for relation extraction.

Term Subsumption tries to extract hierarchical relations between terms by comparing their level of abstraction in a document using conditional probability. The algorithm states that term t_1 is more general than term t_2 if the probability of observing term t_1 in the presence of term t_2 is higher than the probability of observing t_2 when t_1 is in the document [19] ($P(t_1|t_2) > P(t_2|t_1)$).

The **Formal Concept Analysis (FCA)** algorithm builds concept hierarchies/taxonomic relations in ontology learning based on the idea that objects are linked to their features (attributes) in the text. In the OntoGain system [46], the objects are terms and the attributes are defined as the terms' syntactic dependencies in the corresponding dependency tree. It constructs a formal context matrix consisting of object-attributes pairs as the input and measures the conditional probability of the pairs. Then, it extracts the relations above a threshold. The evaluation of relation extraction compared to agglomerative clustering on medical domain shows that FCA is not only more complex ($O(2^n)$), but also less accurate [46] than bottom-up clustering approaches, which iteratively merge clusters and whose complexity is $O(N^2)$. Contento [47] is an ontology construction kit based on FCA that provides the user a bottom up ontology construction in four stages: 1) extraction of data from SPARQL endpoints; 2) generating a FCA lattice; 3) annotating and prune the conceptual lattice; 4) generating the OWL ontology.

Hierarchical Clustering algorithms are usually employed to extract taxonomic relations among concepts. They are categorized into two main groups according to their approach in the cluster construction: agglomerative clustering (bottom up) [48] and divisive clustering (top down). For measuring the similarity among the data, they employ different metrics such as Cosine or Jaccard similarity. Agglomerative clustering combines the most similar elements progressively to reach to a certain stage in which most optimal concepts are present. To compute cluster similarity three approaches can be applied: single linkage, complete linkage and average linkage. Divisive clustering, instead, considers a single cluster which includes all the elements. Then, it iteratively splits large clusters into smaller ones by applying k-means or other clustering algorithms [49].

Foundational Ontology and Reasoner-enhanced axiomatiZAtion (FORZA) [50] is a generic approach for solving some ambiguities while obtaining the ontology and is integrated in the MoKi ontology development tool. Particularly, it aims to assist in acquiring part-whole relationships among objects, also axiom extraction by applying decision diagrams.

Association Rule Mining (ARM) is a data mining approach that is mostly used for extracting non taxonomic relations and patterns among concepts by exploration of the rules in order to predict the co-occurrence of entities. The most popular algorithms for mining association rules are the Apriori method and Frequent pattern (FP) growth.

Axiom Extraction:

Inductive Logic Programming (ILP):

In the last stage of ontology learning ILP is often used for automatic extraction of rules out of schematic axioms utilizing background knowledge and a set of examples which are represented as a logical database of facts [19]. More precisely, given background knowledge B, expressed as a set of predicate definitions, a set of observations consisting of positive examples E+ and negative examples E-, an ILP will construct a predicate logic

formula H such that i) all the possible examples in E+ can be logically derived from B ∧ H, and ii) no negative example in E- can be logically derived from B ∧ H [51].

3.2.4 Evaluation:

After acquiring the ontology, it should be evaluated to be fit to the users' requirements considering the following factors: the lexical correctness of the obtained ontology, its coverage for the concepts, wellness at taxonomic relations and the adequacy of its non-taxonomic relations [19]. However, since the whole process is composed of multiple stages, the evaluation is a complex task. The techniques used for the evaluation can be categorized into four groups according to the kind of target ontologies and the purpose of the evaluation:

Gold standard-based evaluation methods provide a frequent and large-scale assessment of the ontology by comparing the obtained ontology with a standard benchmark called reference ontology which is an ideal model, possibly formalized by experts or acquired from the corpus of that specific domain. The benchmark should be human-created or a reliable standard from a similar domain so that completeness, accuracy and consistency can be truly validated. Golden standard-based methods are also referred to as ontology mapping or ontology alignment.

Application-based (or task-based) methods are task-oriented techniques which evaluate the performance of the obtained ontology on the basis of its outcome in a particular application, without considering its structural properties. They reveal inconsistent concepts and the level of adaptability of the ontology in that domain and measure the compatibility to different ontology learning tools. They evaluate the correctness, coverage, adequacy and wellness of obtained ontology in the desired application.

Data driven-based (or corpus-based) methods measure the coverage of the acquired ontology by comparing target ontology with a particular corpus. To do so, they use domain-specific knowledge sources. They assess similar metrics to golden standard-based techniques such as completeness, conciseness and accuracy. However, instead of a reference ontology they require a domain-specific corpus.

Human or criteria-based evaluation techniques are usually used to select the best ontology among many existing ones. For doing that, several criteria are defined and formulated based on which ontologies are scored and ranked. The major drawback is the high manual cost required in terms of time and effort in the implementation.

3.2.5 Bridging the gap between domain knowledge and ontology engineering:

Usually creating a suitable ontological structure is a difficult task because, firstly, it requires an extensive effort of domain experts and, secondly, available tools for constructing ontologies are often too complex for domain experts who have limited skills in engineering tools such as OWL and RDF. This leads to hinder the process of ontology construction. [52] proposes a holistic approach for constructing a conceptual ontology to support domain experts in ontology authoring. A conceptual ontology combines the human readability and understandability of an ontology, which is domain knowledge independent from the logical formalism, and machine interpretability, which is a logical representation of the ontology. In other words, it provides a descriptive logic understandable for both machine and humans by introducing three integrated components:

1) **Kanga** is a methodology that involves domain experts in authoring the ontology. It covers conceptual aspects written by domain experts and the logical aspects of converting (either manually or automatically) the conceptual ontology into a Semantic Web language such as OWL. The following table presents the steps included:

Steps:	Responsible
1. Identification of the scope, purpose and other requirements of the ontology	domain experts
2. Collecting source knowledge and documents	domain experts
3. Creating a knowledge glossary containing ontology content	domain experts
4. Formal definition of core concepts and relationships using English sentences	ontology engineers (mainly)
5. Conversion of English sentences into OWL.	ontology engineers (mainly)

2) **Rabbit** is a controlled natural language[10] (CNL) to automate the stated process and to encode the knowledge which is going to be converted to OWL. The main aim of Rabbit is providing an understandable platform for domain experts to express not only the axioms, but also the necessary details of the grammar while establishing a well-defined grammar to be translated to OWL systematically.

3) **ROO** (Rabbit to OWL Ontology authoring) is a friendly user interface which guides target users about the phases of constructing an appropriate conceptual ontology in Rabbit CNL and then convert the sentences to a logical form.

The **instance-based learning method** is an approach for extracting structured data from Web pages by comparing new instances to the existing ones based on the idea that structured data are usually created according to some fixed layouts, so unlike classic instance-based learning, [53] provides an approach in which templates can be obtained by learning from a single labeled instance. The requirement is an initial set of some labeled instances, then new instances can be compared to them. Similarity of data is measured by a method called “sufficient match”, which exploits the HTML markup. Two instances are sufficiently similar if a minimum number of tokens in the new instance matches with the prefix and suffix tokens of a labeled instance.

3.3 UML-BASED ONTOLOGY CREATION

UML-based approaches to ontology creation define mechanisms to automatically generate ontologies from UML diagrams through suitable transformations. We can identify two broad categories of techniques:

- Model-driven approaches, in which UML diagrams (usually conforming to some specialized UML profile) are used as a high-level notation for describing the features of ontologies; in this case, ontologies are produced through translation of the UML models into declarations in a suitable ontology description language (usually OWL).

[10] CNL (Controlled Natural Language) is a subset of natural language which is parsable by a computer, and expressive enough for non-specialists

- Techniques in which ontologies are extracted from existing, plain (i.e., not conforming to a specialised profile) UML Class Diagrams.

Model-driven approaches are based on the idea of using techniques from the software engineering domain to help domain experts produce ontologies without having to learn the technicalities of ontology definition. These techniques are based on the following ingredients:

- A high-level modelling notation – typically a UML profile – which is used by domain experts to describe the elements and constraints of the ontology.
- A transformation technique, which is capable of generating declarations in a target ontology description language (e.g., OWL or one of its subsets) from the parsing of the elements of the UML diagrams.

The approach described by Gašević et al. in [54] belongs to this category of works.

Kim and Lee [55] introduce a model-driven approach for the creation of semantically rich descriptions of web services given in terms of the OWL-S language. The approach of [55] relies on a pair of transformations. First, ontological concepts are imported in UML models through a translation of OWL declarations into the elements of a UML Class Diagram. These elements are then used in UML Sequence and Activity Diagrams (enriched with suitable stereotypes) to describe the process executed by the web service. Finally, the UML diagrams are translated into OWL-S descriptions.

Approaches for the **extraction of ontologies** from existing (plain) UML diagrams aim to reuse existing UML models, which already contain codified information about many different domains, to facilitate the creation of formal ontologies. These approaches transform the elements of UML Class Diagrams (classes, associations, attributes, objects, etc.) into corresponding ontological concepts. For example, UML classes are translated into OWL classes, UML attributes are translated into data and object properties, and so on. Examples of such approaches can be found in [56] and [57]

The approaches presented in this section could be used to facilitate the importing of existing data models – such as for example the Transmodel[11] and NetEx[12] – in the Shift2Rail Interoperability Framework.

3.4 DATA MAPPING TECHNIQUES

In the last years, the need to expose relational data on the Web has considerably increased. On the other side, the use of the Resource Description Framework (RDF) [58] to represent data on the Web can ease their integration and retrieval, exploiting the RDF semantics and mechanisms. Moreover, the RDF data representation lets data be accessible both for humans and for machines [59].

The techniques that take an RDB schema and data as an input and produce one or more RDF graphs has given rise to the RDB2RDF Mapping Language (R2RML) as a standard recommendation from the W3C consortium [60].

[11] http://www.transmodel-cen.eu

[12] http://netex-cen.eu

Recently, some researchers have proposed the RDF Mapping Language (RML), i.e. an extension of R2RML to support mappings of data sources in other structured formats, like CSV, XML and JSON [61].

When a reference conceptual model is available, for example in the form of a Web Ontology Language (OWL) [62] ontology, the need arises to use that model to query potentially sources of data, which are semantically homogeneous but syntactically/structurally heterogeneous. This approach is usually referred to as Ontology-based Data Access (OBDA), i.e. a new paradigm to access data sources based on the use of knowledge representation and reasoning techniques [63].

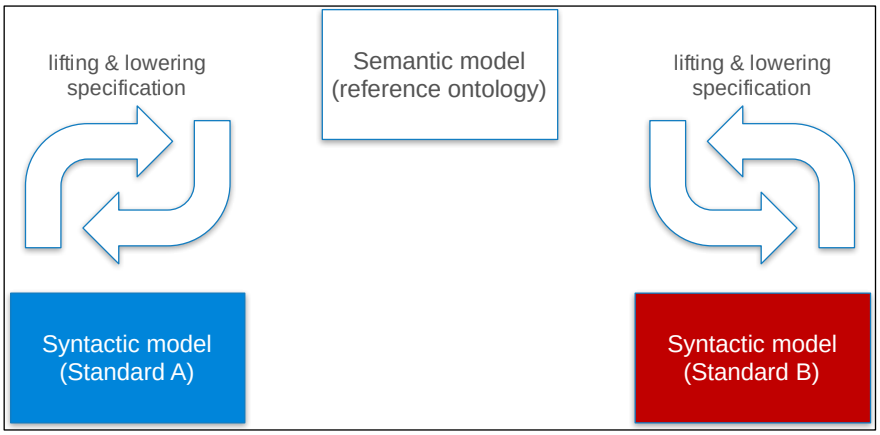

Figure 4: any-to-one approach for semantic interoperability overview

When two or more heterogeneous systems need to interoperate, the any-to-one approach for semantic interoperability has been proposed [64]. In this context, a reference ontology is considered as a conceptual reference point for all involved systems and data are converted between sources and the reference ontology. Since ontologies usually represent a higher level of abstraction with respect to other structured formats, the translation from a data source to the ontology is called “lifting”, while the opposite transformation is called “lowering” [65]. In this sense, mapping from source A to source B can be viewed as a two-step process: lifting from A to the reference ontology and then lowering from reference ontology to B (see Figure 4).

Moreover, the reverse process from an ontological format to its original structure (i.e., the lowering) by means of the same mapping used for the lifting has been investigated (for the CSV format) [66]. The possibility of reversing the conversion process from source A to the ontological format would imply that a single mapping definition is needed, i.e. the lifting mapping for source A, because this mapping would automatically represent also its lowering. Reverse mapping, however, is still an open research problem.

In this way, mapping source A to B, to whatever source N, could be approached as a modular process, made by reusable mapping definition of the i-th source to the reference ontology.

Below is given a brief description for each of the aforementioned techniques.

3.4.1 OBDA

The key idea of OBDA is to provide users with access to the information from the data sources through a three-level architecture, constituted by the ontology, the sources, and the

mapping between the two (see Figure 5). The ontology is a formal description of the domain of interest, and is the heart of the system. Through this architecture, OBDA provides a semantic end-to-end connection between users and data sources, allowing users to directly query data spread across multiple distributed sources.

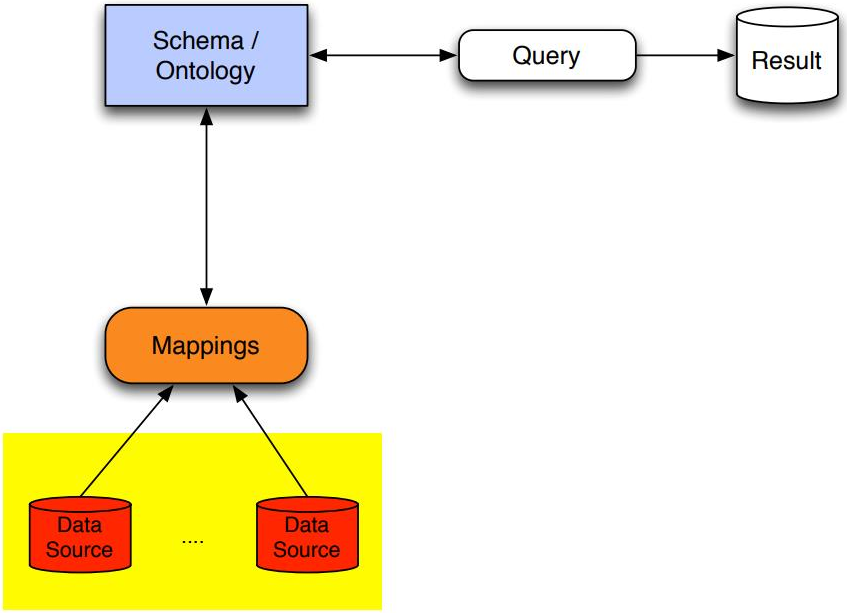

Figure 5: OBDA overview

Using the familiar vocabulary of the ontology, the user formulates queries over the ontology using the SPARQL Protocol and RDF Query Language (SPARQL) [67]; SPARQL queries are then transformed, through the mapping layer, into SQL queries over the underlying relational databases.

Tools implementing an OBDA approach include:

- Karma: https://github.com/usc-isi-i2/Web-Karma
- Ontop: https://github.com/ontop/ontop

3.4.2 R2RML

R2RML is a language for expressing customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice.

Every R2RML mapping is tailored to a specific database schema and target vocabulary. The input to an R2RML mapping is a relational database that conforms to that schema. The output is an RDF dataset, which uses predicates and types from the target vocabulary. The mapping is conceptual: R2RML processors can materialize the output data, offer virtual access through an interface that queries the underlying database, or offer any other access means to the output RDF dataset.

The data consumer essentially can access the (virtual or materialized) RDF data in different ways:

1. Query access, which means the agent issues a SPARQL query against an endpoint exposed by the system and processes the results (typically the result is a SPARQL result set in XML or JSON).
2. Entity-level access, which means the agent performs an HTTP GET on a URI exposed by the system and processes the result (typically the result is an RDF graph).
3. Dump access, which means the agent performs an HTTP GET on dump of the entire RDF graph, for example in Extract, Transform, and Load (ETL) processes.

R2RML mappings are themselves expressed as RDF graphs.

The R2RML specification has a companion that defines a Direct Mapping (DM) from relational databases to RDF [68]. In the DM of a database, the structure of the resulting RDF graph directly reflects the structure of the database, the target RDF vocabulary directly reflects the names of database schema elements, and neither structure nor target vocabulary can be changed. DM can be used when a pre-existing target vocabulary is not available. With R2RML, on the other hand, a mapping author can define highly customized views over the relational data.

Tools implementing R2RML include:

- Db2triples - R2RML and DM implementation: https://github.com/antidot/db2triples
- R2RML parser: https://github.com/nkons/r2rml-parser
- Morph - RDB2RDF tool using R2RML mappings: https://github.com/jpcik/morph

3.4.3 RML

RML [61] is a generic scalable mapping language defined to express rules that map data in heterogeneous structures and serializations to the RDF data model. Such mappings describe how existing data can be represented using the RDF data model. RML is based on and extends R2RML, which is defined to express customized mappings only from relational databases.

An RML mapping is not tailored to a specific database schema as an R2RML mapping, but it can be defined for data in any other source format (currently defined for data sources of structured formats such as CSV, XML, and JSON). RML keeps the mapping definitions as in R2RML, but it excludes its database-specific references from the core model of the mapping definition. RML provides a generic way of defining the mappings that is easily transferable to cover references to other data structures. Thus, RML is a generic approach combined with case-specific extensions, but it always remains backward-compatible with R2RML as relational databases represent such a specific case.

The input to an RML mapping can be any data source. The output is an RDF dataset that uses predicates and types from the target vocabulary. In RML, like in R2RML, the mapping definitions are expressed as RDF graphs[13].

Tools implementing RML include:

- RML mapper: https://github.com/RMLio/rmlmapper-java

[13] http://rml.io/index.html

- RML editor: http://rml.io/RMLeditor
- RML validator: https://github.com/RMLio/RML-Validator
- RML engine: https://github.com/carml/carml

3.4.4 Reversing RML

After defining an RML mapping for the lifting transformation of a data source, it would be very useful to “reverse” the direction of the transformation, in order to realize the lowering process, converting back the data expressed in the reference ontology to the original source format. However, this reverse mapping is not always possible and can pose a number of issues making it a very challenging task to accomplish; investigations exist that studied the assumptions and constraints under which the reverse operation is feasible.

In the case of column-based data sources, some work [66] was done to perform the reverse process for transforming an RDF dataset into its CSV tabular structure, through the use of the same RML mapping document that was used to generate the set of RDF triples.

To reach the goal a pair of assumptions have been made: the first is related to the set of RML mapping rules used to expose the CSV data source in RDF; the second concerns the implicit cardinality constraints of the associations between the columns of the CSV data source. Informally, the assumptions are as follows:

1. Dependency Tree Assumption: the graph underlying the mapping rules is one n-ary tree, i.e. it will have:
    a. only one vertex, the root, that does not have incoming edges,
    b. one or more vertices, leaves, that do not have outgoing edges,
    c. there is at most one path (always starting from the root node) that connects two nodes and
    d. each node has no more than n children.
2. Implicit Cardinality Restrictions: the original CSV data source, from which the RDF dataset was generated, has associations between columns with only 0:0 or 1:1 cardinality constraints.

Those assumptions seem very restrictive; however, they are verified when the CSV data source has a structure containing at least one column with unique value that could be used as key and the RML rules in the mapping are extended to keep links between RDF triples that refer to the same row [69].

This proposed approach is implemented in the tool available at: https://bitbucket.org/carloallocca/rml2csv

3.5 SEMANTIC DISCOVERY

With the term “Semantic discovery” we mean the description of resources by users in a catalogue via controlled vocabularies, and the techniques that can be used to let users access such knowledge. Such resources can be (among the others) datasets, data models or service descriptions.

Resource descriptions are the key element to let users discover what is contained in an asset management catalogue. Different vocabularies have been proposed over the years to describe metadata, and in the following we will describe some of the most commonly used ones.

We will then analyse some techniques that have been proposed to allow users to query an RDF repository in a controlled way via Web API. Such techniques allow exposing only a subset of the whole information contained in a repository in order to comply with different user authorization profiles and to streamline checking the users’ behaviour.

3.5.1 Catalogue metadata vocabularies

Data Catalogue Vocabulary (DCAT) [70] is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. The basic idea of DCAT is to model a Catalogue, which is composed by several Datasets, each one having different embodiments, as depicted in Figure 6.

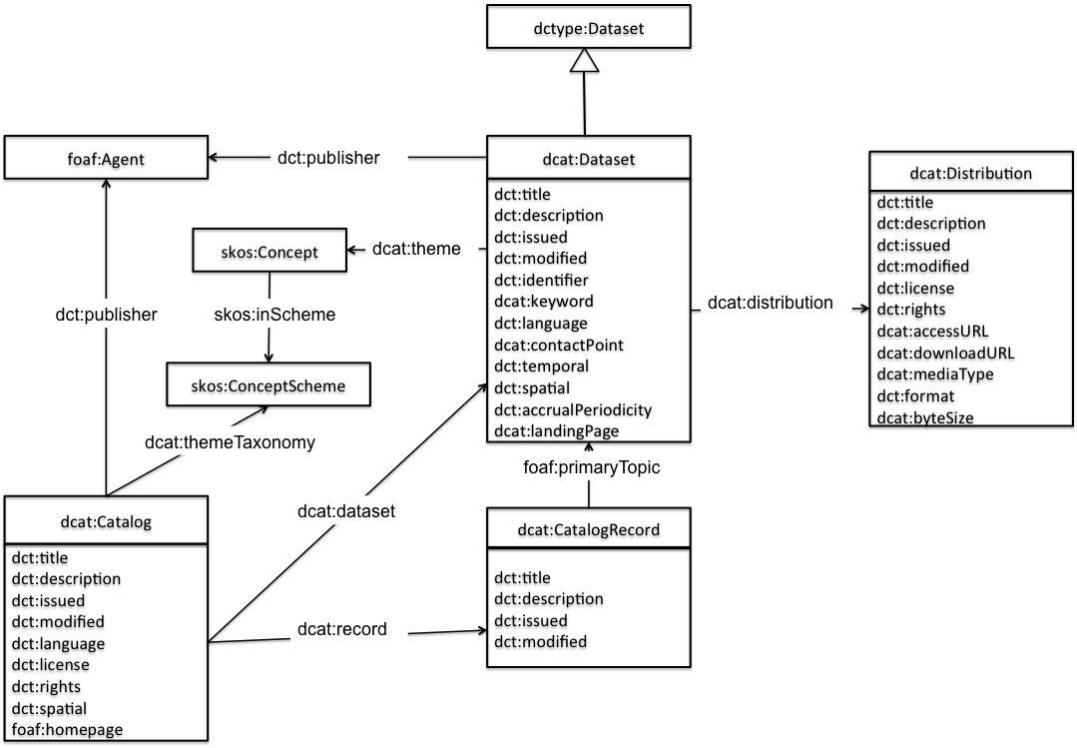

Figure 6 DCAT data model

DCAT allows defining different Application Profiles, which are specifications that re-use terms from one or more base standards, adding more specificity by identifying mandatory, recommended and optional elements to be used for a specific application, as well as recommendations for controlled vocabularies to be used.

The DCAT Application Profile for data portals in Europe (DCAT-AP) is a specification based on the DCAT developed by W3C.

This application profile is a specification for metadata records to meet the specific application needs of data portals in Europe while providing semantic interoperability with other applications on the basis of reuse of established controlled vocabularies (e.g., EuroVoc) and mappings to existing metadata vocabularies (e.g., Dublin Core, SDMX, INSPIRE metadata).

Since it is focused on data portals, the DCAT-AP use case is the publication of generic datasets without further interest in a fine-grained definition of the purpose or type of the specific dataset.

Asset Description Metadata Schema (ADMS) [71] is a profile of DCAT, used to describe semantic assets (or just 'Assets'), defined as highly reusable metadata (e.g., XML schemata, generic data models) and reference data (e.g., code lists, taxonomies, dictionaries, vocabularies) that are used for eGovernment system development. Compared to DCAT-AP, this profile adds the possibility to differentiate Assets into homogeneous categories (Asset Types). This DCAT profile is therefore highly relevant for the Asset Manager component of the IF, because it provides a schema for the publication of the minimum set of information that can allow the publication of information about an asset in the Catalogue.

An ADMS *Asset* is an entity reflecting the intellectual content of an Asset and represents those characteristics that are independent of its physical embodiment. This abstract entity combines the FRBR[14] entities work (a distinct intellectual or artistic creation) and expression (the intellectual or artistic realization of a work).

Each *Asset* belongs to one or more *Asset Types,* which are classifications of assets according to a controlled vocabulary. Each *Asset* is then made available via a *Publisher* organization. The physical embodiment of an Asset is called an *Asset Distribution*. A specific Asset may have zero or more Distributions. A Distribution is typically a downloadable computer file (but in principle it could also be a paper document or API response) that implements the intellectual content of an Asset. A Distribution is associated with one and only one Asset, while all Distributions of an Asset share the same intellectual content in different physical formats.

3.5.2 Service descriptors

Service orientation has increasingly been adopted as one of the main approaches for developing complex distributed systems from reusable components called services. Just as semantics has brought significant benefits to search, integration, and analysis of data, it is also seen as a key to achieving a greater level of automation to service-orientation. This led to research and development, as well as standardization efforts on Semantic Web Services. Such activities have involved developing conceptual models or ontologies, algorithms, and engines that could support machines in semi-automatically or automatically discovering, selecting, composing, orchestrating, mediating, and executing services, with the aim to use semantic technologies to help with the following tasks:

- *discovery*, which is the task of matching known Web services against a user goal and returning the services that can satisfy that goal;

[14] https://en.wikipedia.org/wiki/Functional_Requirements_for_Bibliographic_Records

- *ranking*, which orders the discovered services based on user requirements and preferences so the best service can be selected;
- *composition*, which puts together multiple services when no single service can fulfil the whole goal;
- *invocation*, which communicates with a specific service to execute its functionality;
- and lastly *mediation*, which resolves any arising heterogeneities.

The initial attempts in Semantic Web Services were aimed at providing the full set of automated tasks, and resulted in OWL-S [72] and Web Service Modelling Ontology (WSMO) [73] initiatives. Such attempts anyhow failed due to the effort in providing an effective and cost-saving environment for the implementation of semantically-enabled services. WSMO in particular provided a very powerful language (WSML [74]) to describe pre-conditions, input, output and effects of a web service operation. The metamodel was composed by Web Services, Mediators, Ontologies and Goals, as depicted in Figure 7.

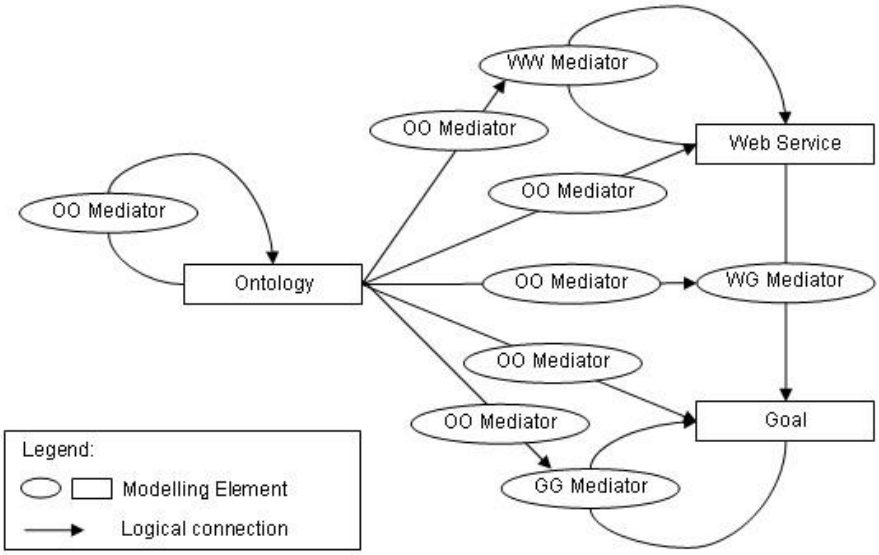

Figure 7. WSMO metamodel

Instead of directly invoking a Web Service operation, the user had to define a Goal, which was to be fulfilled by the Semantic Web Services Engine by orchestrating and executing services in a complex process. The same approach was then adapted to W3C Semantic Web stack in WSMO-Lite, abandoning the dedicated WSML language in favour of RDF and RIF (Rule Interchange Format [75]), as depicted in Figure 8.

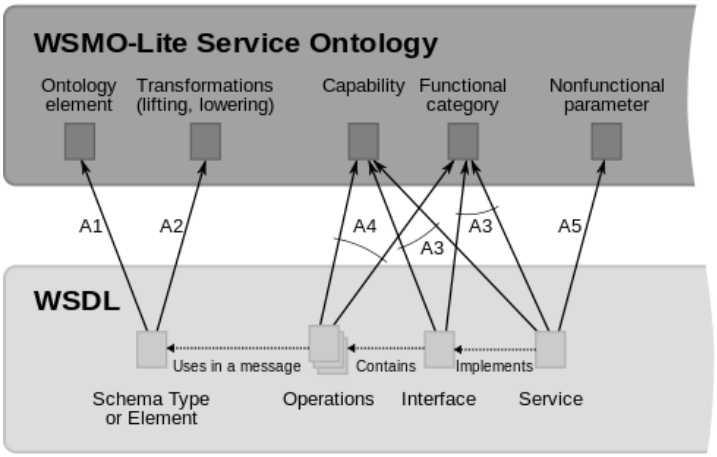

Figure 8. How WSMO-Lite concepts can be referenced in WSDL

While Web Services have a well-defined and accepted set of standards (SOAP and WSDL), there is no generally accepted machine-processable description language for RESTful Web services. hRESTS [76] was an attempt to define a microformat to embed structured service descriptions inside the HTML pages describing a service documentation. hRESTS was made up of a number of HTML classes directly corresponding to the various parts of the service model depicted in Figure 9.

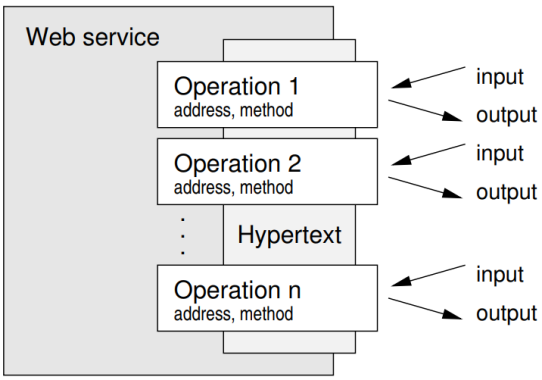

Figure 9. hRESTS model of a Web Service

The interaction of a client with a RESTful service is seen as a series of *Operations* where the client sends a request to a resource (using one of the HTTP *methods* GET, POST, PUT or DELETE), and receives a response that may link to further useful resources. A RESTful Web service has a number of operations, each with potential *Inputs* and *Outputs*, and a hypertext graph structure where the outputs of one operation may link to other operations.

While hRESTS provided the technical means to add structured descriptions of a RESTful service, it lacked the expressivity to link to an existing formal ontology. MicroWSMO [77] was an extension to hRESTS providing such link, with the same purpose of SAWSDL [78] in the Web Services stack, as depicted in Figure 10.

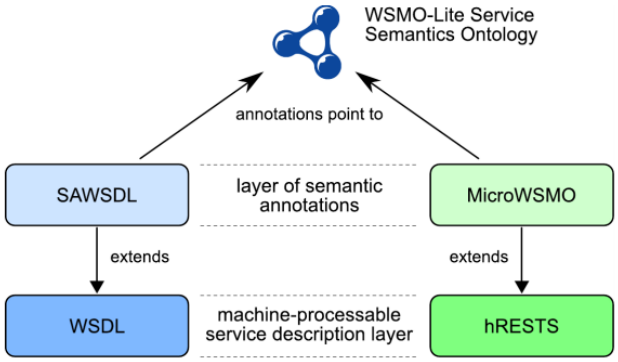

Figure 10. Similarities between WSDL and hRESTS and between SAWSDL and MicroWSMO

Mimicking SAWSDL specifications, MicroWSMO introduced the following three types of HTML link relations:

- *model* indicates that the link is a model reference to a concept expressed in a WSMO-Lite ontology;
- *lifting* and *lowering* denote links to the respective data transformations.

This enabled to extend the use the WSMO stack to RESTful services, and allowed for the provision of automatic discovery, orchestration and execution of services.

Such richness in expressivity came anyway at a cost, since the effort to create a working description of a service was comparable to the effort spent in implementing the service itself. Moreover, it did not deal with trust, since it was not possible to force the engine to restrict the execution to well-known services. Once the goal was sent to the engine, there was no way of driving the computation. Such heavyweight approaches aimed at automatic orchestration and execution were therefore abandoned in favour of lighter approaches tailored to ease discovery and selection.

Lightweight approaches to semantic descriptions focus only on service discovery, thus leaving out ranking based on non-functional properties, automatic composition and execution. They therefore offer an aid to the developer during the task of finding services to be integrated in an application.

One of such approaches is Hydra [79], which has been proposed as a lightweight vocabulary to create hypermedia-driven RESTful APIs. The basic idea behind Hydra is to provide a vocabulary which enables a server to advertise valid state transitions to a client. A client can then use this information to construct HTTP requests which modify the server's state so that a certain desired goal is achieved. All the information about the valid state transitions is exchanged in a machine-processable way at runtime instead of being hardcoded into the client at design time. As can be noticed in Figure 11, the Hydra vocabulary revolves around the main "Operation" concept, which is used to link input and output to ontology concepts.

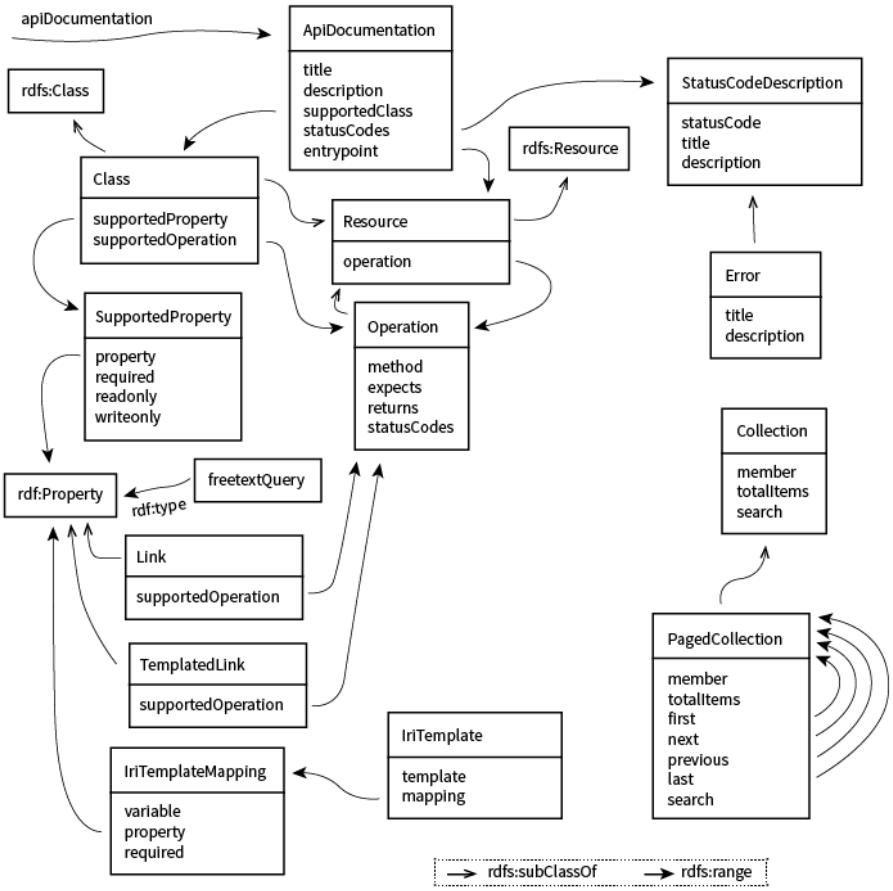

Figure 11. Hydra ontology

Hydra does not reuse already existing RESTful API description specification, and it rather proposes the whole stack from the vocabulary to the generation of human-readable API documentation.

A different approach, proposed in [80], consists in using the existing extension points defined in the OpenAPI specifications (OAS) to add links to ontological concepts and relations. The article describes a highly effective way of adding semantics to machine-readable descriptions without disrupting the existing toolchain. The article envisions the usage of the following attributes to represent a mapping to existing concepts in an ontology:

- *x-refersTo*: it specifies the concept in a semantic model that best describes an Open API Specification (OAS) element.
- *x-kindOf*: it specifies a specialization that exists between an OAS element and a concept in a semantic model. The property is mainly used to declare that a concept in a semantic model is too generic to describe the specified model, whereas a more specific subtype (if it exists) should be considered more appropriate.
- *x-mapsTo*: it indicates that an OAS element is semantically similar to another OAS element.
- *x-collectionOn*: it indicates that a model describes a collection over a specific property.
- *x-onResource*: it denotes that the specific Tag Object refers to a resource described by a Schema Object.

- *x-operationType*: the property is used to provide semantic information on the type of operation. The subtypes of the Action concept in the Schema.org vocabulary can be used as values of the property.

The elements above are then exploited to create instances of the concepts of the OpenAPI ontology, whose model is represented in Figure 12.

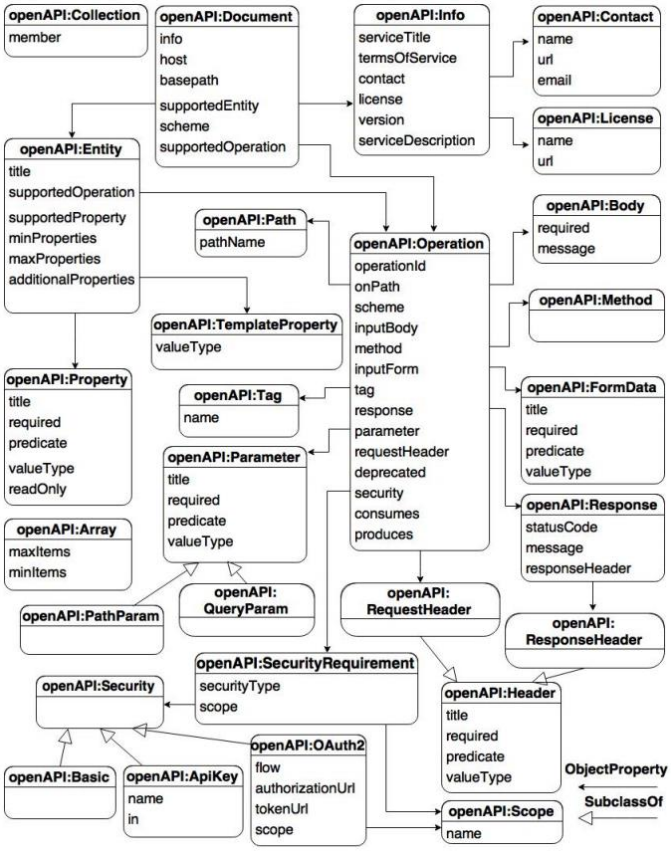

Figure 12. OpenAPI ontology

3.5.3 GraphQL

Today's web and mobile applications are often data-driven and require large sets of data combining related resources. Accessing those data by using a REST-based API often requires us to do multiple round-trips to collect everything that is needed. As REST, GraphQL is an API design architecture, but with a different approach which is much more flexible. The main and most important difference is that GraphQL is not dealing with dedicated resources. The main feature of GraphQL is to be able to send queries rather than asking for the representation of a resource as a whole.

GraphQL requires the definition of a data schema which tells the users which data they can query. The two main selling points of this technology, namely the query capabilities and the schema, are quite similar to what SPARQL and ontologies are offering in the Semantic Web stack. In a sense, GraphQL can be seen as a simplified way to define a query over a graph with respect to SPARQL, with the main difference being that SPARQL does not assume a fixed schema or vocabulary. In certain applications, where the data schema is well known

and fixed, GraphQL can be much easier to understand with respect to SPARQL, and more efficient in terms of bandwidth with respect to REST.

With the aim of allowing the usage of GraphQL technology over RDF data, GraphQL-LD has been proposed as an extension to GraphQL to allow queries with a JSON-LD context [81]. The approach consists of an algorithm that translates GraphQL-LD queries to SPARQL algebra as depicted in Figure 13. This allows such queries to be used as an alternative input to SPARQL engines.

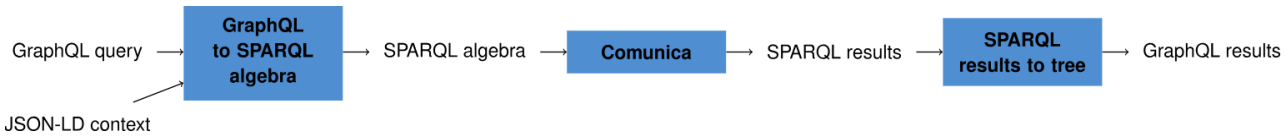

Figure 13. GraphQL rewriting in GraphQL-LD

HyperGraphQL [82] is another attempt to expose access to RDF sources through GraphQL queries and emit results as JSON-LD. The main difference with the GraphQL-LD approach is that HyperGraphQL is implemented by means of an intermediary service sitting between the GraphQL client and the RDF repository, while GraphQL performs a query translation and directly executes the native query.

3.5.4 Query templates for data access

GraphQL is emerging as an efficient technology which overcomes some of the limitations of the REST approach. Its success is showing that exposing a query endpoint and letting a user ask for whatever data he/she wants can lead to bandwidth-friendly applications. Anyway, such approach has a few drawbacks related to security and how the application owner can monitor what the users are doing. Securing a query endpoint is much more difficult than securing an API endpoint, because if the GraphQL schema is complex enough there is the possibility to overload the application with complex and long queries and obtain a denial of service. Moreover, using an “open” query endpoint means that it is much more difficult to do usage statistics and understand what the users are asking.

Another approach to fill the gap between graph repositories and Web APIs is to wrap SPARQL queries and expose them as Web APIs. Template engines are used to let developers express a query with parameters. Such parameters are then exposed as the Web API operation parameters. Users call the API with the usual means, and they are not even aware that the operation is implemented by means of a SPARQL query. Such approach allows the application developer to clearly control how data is accessed, because the users are forced to use such Web APIs, and the query endpoint is not exposed.

Inspired by this, [83] presents *grlc*, a lightweight server that translates SPARQL queries curated in GitHub repositories to Linked Data APIs on the fly. The same approach, heavily influenced by *grlc*, has been implemented with the concept of “Exploration API” in the IT2Rail Asset Manager. Instead of using GitHub as a repository for template SPARQL queries, the IT2Rail Asset Manager allows publishing such APIs as *assets* inside the application. Whenever an Exploration API is published in the system, the API is exposed so that users can query the RDF repository managed by the Asset Manager.

4. DISCUSSION

This deliverable analysed the state of the art regarding several topics of interest for the SPRINT project from the point of view of semantic technologies, in particular for what concerns the integration of processes. In particular, it focused on the following issues:

1. The results of past S2R projects regarding the creation and management of sematic technologies for the IF.
2. The creation and maintenance of ontologies, possibly through learning mechanisms.
3. The mapping of data from one format to another.
4. The discovery of services.

The first issue was analysed in Section 2. The analysis showed that the Asset Manager developed in the IT2Rail project conceptually provided three very important concepts, namely the ability to configure multiple asset types, the possibility to define asset lifecycles and the role of the Asset Manager itself as the official publishing tool for data residing in the Semantic Graph. Such features can now be further inspected and extended, e.g. by investigating the usage of much more powerful process languages to define asset lifecycles. The role of the Asset Manager as the catalogue of all the assets pertaining to an IF node paves also the way for a wider usage of this tool to improve the automation features of the IF.

The GOF4R project highlighted a number of governance problems that directly impact assets handled by the IF, in particular ontologies. Indeed, the ST4RT project proposed a number of modifications to the ontology developed in the IT2Rail project, but there is still not a process in place to harmonize and merge the different versions of the ontology. Also, an analysis of the annotation process, first introduced in the IT2Rail project and then extended in the ST4RT project to map data from one format to another, showed that the process is still rather labor-intensive, and automated mechanisms should be developed to ease the burden on domain experts.

Then, Sections 3.1, 3.2 and 3.3 surveyed a number of techniques that could help address the issues of ontology development highlighted above.

As discussed in Section 3.1, there are two aspects that need to be considered for the ontology development activities to be carried out in the context of the SPRINT project. On the one hand, an appropriate ontology engineering methodology needs to be selected. In our case, we have decided to make use of one of the most complete ontology engineering methodologies available in the state of the art, which is especially focused on the development of networks of ontologies in a collaborative setting, such as the ones that are expected to be built in SPRINT. As we have discussed, much of the ontology development to be done will be based on the reuse of non-ontological resources, hence we will focus on making use of scenario number 2 from the methodology. On the other hand, a set of tools that can provide support to the methodology that has been selected needs to be considered. In our case, we have decided to select a state-of-the-art ontology development tool, such as Protégé, widely used in the state of the art for OWL ontology development, and the OWL files that will be developed will be maintained using a state-of-the-art source code management platform (GitHub), which will be connected with Ontoology, so as to provide support to the full ontology development lifecycle.

One of the goals of the SPRINT project is to improve on the techniques developed in the ST4RT project by automating the process with which the concepts in legacy data formats are mapped to the reference ontology. Mechanisms for automatically learning ontologies from instances of data could help in this regard, so Section 3.2 provided a general overview of the automatic ontology learning process. It weighed the role of pre-processing techniques before applying any ontology learning algorithm to achieve satisfactory results for both structured and unstructured case studies. Also, it highlighted the importance of evaluation methods and their state-of-the-art techniques according to the purpose of the learning. Moreover, it presented the stages involved in the process of ontology learning including term, relation and finally axiom extraction, and it explained some of the current tools and methodologies for each stage. Then, it categorised the main existing evaluation methodologies and discussed their usage, advantages and drawbacks. To bridge the gap between ontology engineering and domain experts who do not have the required engineering skills, a platform with conceptual aspects is discussed; finally, the section introduced an instance-based learning method specific for structured text.
Since the data that needs to be mapped to the reference ontology is most likely captured through a structured format, for the research purposes of the SPRINT project this latter, instance-based, learning method, and tools such as OntoLearn, which are more suitable for learning from structured text, seem the best candidates to be the basis (or the inspiration) for future developments in the project.

Section 3.3 pointed out a few techniques for building OWL ontologies from existing UML models. These techniques could be useful if the formats to be targeted, for example in a conversion, had a UML description in terms of Class Diagrams. However, currently it does not seem that a similar scenario is relevant within the SPRINT project.

Section 3.4 analyzed techniques that could be used to improve the data mapping techniques that are at the core of the converter technology developed in the ST4RT project. As Section 3.4 described, there is a plethora of solutions concerning mapping different data formats to ontologies, thus performing lifting. On the opposite side of the transformation pipeline, the problem of extracting data from an RDF source to populate data schemas in different formats (lowering) has received less attention by researchers. Among the different techniques to implement lifting, both the ST4RT solution and the RML language can be considered good candidates to be included in the SPRINT converter. The former offers a set of Java annotations that cannot be found in any other RDF toolkit. Such annotations allow obtaining complex mappings which are absolutely required when dealing with real-word complex transportation standards (as described in Section 2.2). RML is also a good candidate because it does not require developers to modify existing source code. This could be a benefit, since full access to source code is not always granted, and also because the requirement of having a Java class representation of the whole standard to be mapped could slow down the adoption of the solution.

Regarding lowering RDF data to legacy data formats, apart from the initial experiments in understanding the possibility to “reverse” an RML lifting mapping, the best candidate to be included in the SPRINT converter is again the ST4RT technology. Lowering is a complex task, possibly much more complex than lifting. From a “topological” point of view, during lifting, a set of tables (in case of database mappings) or an information tree (in case of an XML mapping) are mapped onto a graph structure, which is a very generic data structure. When performing lowering, instead, a very generic data structure must be mapped in a stricter data structure. This leads to the necessity of a very specific mapping which would

take care of both the information mapping and the structural mapping. A language offering such features does not yet exist, therefore the annotation-based approach exploited in ST4RT can be considered the only existing approach alternative to manually-coded data conversion. Such approach can therefore be used in SPRINT and tested with the new converter architecture which will tackle scalability.

Semantic asset discovery techniques like grlc and GraphQL, described in Section 3.5, deal with mixing APIs and query capabilities. GraphQL can be used to expose a query endpoint so that users can freely choose which data to obtain, whereas grlc allows using common HTTP technologies to allow users to access a pre-selected set of queries. The possibility to wrap a query to the semantic graph and expose it as a Web API could lead to the streamline of some coding activities which are currently required to implement Resolvers in the IF.

5. REFERENCES

[1] IT2Rail Consortium, "www.it2rail.eu," [Online].

[2] ST4RT Consortium, "www.st4rt.eu," [Online].

[3] GOF4R Consortium, "www.gof4r.eu," [Online].

[4] ISO/IEC, *10746-3:2009 Information technology -- Open distributed processing -- Reference model: Architecture*, 2019.

[5] M. Fernández-López, A. Gómez-Pérez and N. Juristo, "METHONTOLOGY: From ontological art towards ontological engineering," in *Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97)*, 1997.

[6] S. Staab, R. Studer, H.-P. Schnurr and Y. Sure, "Knowledge processes and ontologies," *IEEE Intelligent Systems*, vol. 16, no. 1, pp. 26 - 34, Jan-Feb 2001.

[7] H. Pinto, S. Staab and C. Tempich, "DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolvInG," in *16th European Conference on Artificial Intelligence (ECAI 2004)*, 2004.

[8] A. Gómez-Pérez, M. Fernández-López and O. Corcho, "The NeOn Methodology for Ontology Engineering," in *Ontology engineering in a networked world.*, Berlin, Heidelberg, 2012.

[9] M. Suárez-Figueroa, A. Gómez-Pérez, E. Motta and A. Gangemi, "Introduction: Ontology Engineering in a Networked World," in *Ontology Engineering in a Networked World*, 2012.

[10] M. Suárez-Figueroa, A. Gómez-Pérez and M. Fernández-López, "The NeOn Methodology framework: A scenario-based methodology for ontology development," in *Applied ontology*, 2015.

[11] V. Presutti, E. Blomqvist, E. Daga and A. Gangemi, "Pattern-Based Ontology Design. In," *Ontology Engineering in a Networked World.*, pp. 35-64, 2012.

[12] L. Halilaj, N. Petersen, I. Grangel-Gonzalez, C. Lange, S. Auer, G. Coskun and S. Lohmann, "Vocol: An integrated environment to support version-controlled vocabulary development," in *In European Knowledge Acquisition Workshop*, 2016.

[13] C. Ghidini, B. Kump, S. Lindstaedt, N. Mahbub, V. Pammer, M. Rospocher and L. Serafini, "Moki: The enterprise modelling wiki," in *European Semantic Web Conference*, 2009.

[14] A. Alobaid, D. Garijo, M. Poveda-Villalon, I. Santana-Perez, A. Fernández-Izquierdo and O. Corcho, "Automating ontology engineering support activities with OnToology," *Journal of Web Semantics*, 9 Oct 2018.

[15] T. Tudorache, J. Vendetti and N. Noy, "Web-Protege: A Lightweight OWL Ontology Editor for the Web," *OWLED*, vol. 432, p. 2009, Oct 2008.

[16] C. Basca, S. Corlosquet, R. Cyganiak, R. Fernández and T. Schandl, "Neologism – Easy Vocabulary Publishing," in *4th Workshop on Scripting for the Semantic Web*, 2008.

[17] A. Stellato, S. Rajbhandari, A. Turbati, M. Fiorelli, C. Caracciolo, T. Lorenzetti, J. Keizer and M. T. Pazienza, "Vocbench: A web application for collaborative development of multilingual thesauri," in *European Semantic Web Conference*, 2015.

[18] P. Buitelaar, P. Cimiano and B. Magnini, "Ontology learning from text: methods, evaluation and applications," p. 2005.

[19] M. N. Asim, M. Wasim, M. U. G. Khan, W. Mahmood and H. M. Abbasi, "A survey of ontology learning techniques and applications," *Database*, p. bay101, 2018.

[20] E. Brill, “A simple rule-based part of speech tagger,” in *Proceedings of the third conference on Applied natural language processing*, Association for Computational Linguistics, 1992, pp. 152--155.

[21] E. Roche and Y. Schabes, “Deterministic part-of-speech tagging with finite-state transducers,” *Computational linguistics 21*, pp. 227-253, 1995.

[22] D. Sanchez and A. Moreno, “Creating ontologies from Web documents,” vol. 113, pp. 11-1.

[23] E. Charniak, “Statistical techniques for natural language parsing,” *AI magazine*, p. 33, 1997.

[24] D. Lin, “PRINCIPAR---An Efficient, broad-coverage, principle-based parser,” *arXiv preprint cmp-lg/9407024*, 1994.

[25] D. Lin, in *31st annual meeting of the association for computational linguistics*, 1993.

[26] D. Klein and C. D. Manning, “Accurate unlexicalized parsing,” in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*.

[27] T. Bergmanis and S. Goldwater, “Context sensitive neural lemmatization with lematus,” 2018, pp. 1391--1400.

[28] K. Oflazer and I. Kuruoz, “Tagging and morphological disambiguation of Turkish text,” in *Proceedings of the fourth conference on Applied natural language processing*, Association for Computational Linguistics, 1994, pp. 144--149.

[29] N. Ezeiza, I. Alegria, J. M. Arriola, R. Urizar and I. Aduriz, “Proceedings of the 17th international conference on Computational linguistics-Volume 1,” pp. 380--384, 1998.

[30] D. Z. Hakkani-Tur, K. Oflazer and G. Tur, “Statistical morphological disambiguation for agglutinative languages,” *Computers and the Humanities*, pp. 381--410, 2002.

[31] T. Erjavec and S. Dvzeroski, “Machine learning of morphosyntactic structure: Lemmatizing unknown Slovene words,” *Applied Artificial Intelligence*, pp. 17--41, 2004.

[32] G. Chrupal a, “Simple data-driven context-sensitive lemmatization,” *Procesamiento del lenguaje natural*, pp. 121-127, 2006.

[33] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Laubli, B. Antonio, V. Miceli and J. Mokry, “Nematus: a toolkit for neural machine translation,” *arXiv preprint arXiv:1703.04357*, 2017.

[34] J. Petit, J.-C. Boisson and F. Rousseaux, “Discovering cultural conceptual structures from texts for ontology generation,” in *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2017, pp. 0225--0229.

[35] A. Hippisley, D. Cheng and K. Ahmad, “The head-modifier principle and multilingual term extraction,” *Natural Language Engineering*, vol. 11, no. 2, pp. 129-157.

[36] D. Faure and C. Nedellec, “Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system ASIUM,” in *International Conference on Knowledge Engineering and Knowledge Management*, 1999.

[37] A. P. Sheth, K. Gomadam and A. H. Ranabahu, “Semantics enhanced services: Meteor-s, SAWSDL and SA-REST,” *Bulletin of the Technical Committee on Data Engineering*, vol. 31, no. 3, p. 8, 2008.

[38] K. Frantzi, S. Ananiadou and H. Mima, “Automatic recognition of multi-word terms:. the c-value/nc-value method,” vol. 3.

[39] P. Velardi, S. Faralli and R. Navigli, “Ontolearn reloaded: A graph-based algorithm for taxonomy induction,” *Computational Linguistics*, vol. 39, no. 3, pp. 665-707, 2013.

[40] R. Navigli and P. Velardi, “Semantic interpretation of terminological strings,” 2002.

[41] B. Frikh, A. S. Djaanfar and B. Ouhbi, “A Hybrid Method for Domain Ontology Construction from the Web.,” 2011.

[42] T. K. Landauer, P. W. Foltz and D. Laham, “An introduction to latent semantic analysis,” *Discourse processes,* vol. 25, pp. 259-284, 1998.

[43] L. Karoui, M.-A. Aufaure and N. Bennacer, “Contextual Concept Discovery Algorithm,” in *FLAIRS Conference,* 2007.

[44] M. Ciaramita, A. Gangemi, E. Ratsch, J. Saric and I. Rojas, “Unsupervised learning of semantic relations between concepts of a molecular biology ontology.,” in *IJCAI' 05 Proceedings of the 19th international joint conference on Artificial intelligence,* Edinburgh, Scotland, 2005.

[45] R. Snow, D. Jurafsky and A. Y. Ng, “Learning syntactic patterns for automatic hypernym discovery,” in *Advances in neural information processing systems,* 2005.

[46] E. Drymonas, K. Zervanou and E. G. Petrakis, “Unsupervised ontology acquisition from plain texts: the OntoGain system,” in *International Conference on Application of Natural Language to Information Systems,* 2010.

[47] E. Daga, M. d'Aquin, A. Gangemi and E. Motta, “Bottom-up ontology construction with Contento,” in *CEUR Workshop Proceedings,* 2015.

[48] M. L. Zepeda-Mendoza and O. Resendis-Antonio, “Hierarchical agglomerative clustering,” *Encyclopedia of Systems Biology,* pp. 886-887, 2013.

[49] R. Ragunath and N. Sivaranjani, “Ontology based text document summarization system using concept terms,” *ARPN Journal of Engineering and Applied Sciences,* vol. 10, no. 6, pp. 2638-2642, April 2015.

[50] C. M. Keet, M. T. Khan and C. Ghidini, “Ontology authoring with FORZA,” in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management,* 2013.

[51] I. Bratko and S. Muggleton, “Applications of inductive logic programming,” *Communications of the ACM,* vol. 38, no. 11, pp. 65-70, 1995.

[52] R. Denaux, C. Dolbear, G. Hart, V. Dimitrova and A. G. Cohn, “Supporting domain experts to construct conceptual ontologies: A holistic approach,” *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 9, no. 2, pp. 113-127, 2011.

[53] Y. Zhai and B. Liu, “Extracting web data using instance-based learning,” in *International Conference on Web Information Systems Engineering,* 2005.

[54] D. Gašević, D. Djurić and V. Devedžić, “MDA-based Automatic OWL Ontology Development,” *International Journal for Software Tools for Technology Transfer,* vol. 9, pp. 103-117, 2007.

[55] I. Kim and K. Lee, “A Model-Driven Approach for Describing Semantic Web Services: From UML to OWL-S,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 39, no. 6, pp. 637-646, 2009.

[56] Z. Xu, Y. Ni, W. He, L. Lin and Q. Yan, “Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach,” *World Wide Web,* vol. 15, p. 517–545, 2012.

[57] J. Zedlitz, J. Jorke and N. Luttenbe, “From UML to OWL 2,” in *Knowledge Technology (KTW 2011),* 2012.

[58] G. Klyne, J. Carroll and B. McBride, “RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation,” February 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/.

[59] S. Auer, L. Feigenbaum, D. Miranker, A. Fogarolli and J. Sequeda, “Use Cases and Requirements for Mapping Relational Databases to RDF. W3C Working Draft,” June 2010. [Online]. Available: https://www.w3.org/TR/rdb2rdf-ucr/.

[60] S. Das, S. Sundara and R. Cygania, “R2RML: RDB to RDF Mapping Language. W3C Recommendation,” September 2012. [Online]. Available: https://www.w3.org/TR/r2rml/.

[61] A. Dimou, M. V. Sande, P. Colpaert, E. Mannens and R. V. d. Walle, “Extending R2RML to a Source-independent Mapping Language for RDF,” in *International Semantic Web Conference (Posters & Demos)*, 2013.

[62] B. Motik, P. Patel-Schneider and B. Parsia, “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) W3C Recommendation,” December 2012. [Online]. Available: https://www.w3.org/TR/owl2-syntax/.

[63] A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini and R. Rosati, “Linking data to ontologies,” *Journal on data semantics X*, pp. 133-173, 2008.

[64] A. Carenini, U. Dell'Arciprete, G. Stefanos, K. P. M.M., M. Rossi and S. Riccardo, “ST4RT–Semantic Transformations for Rail Transportation,” in *Transport Research Arena TRA*, 2018.

[65] J. Farrell and H. Lausen, “Semantic Annotations for WSDL and XML Schema W3C Recommendation,” August 2007. [Online]. Available: https://www.w3.org/TR/sawsdl/.

[66] C. Allocca and A. Gougousis, “A Preliminary Investigation of Reversing RML: From an RDF dataset to its Column-Based data source,” *Biodiversity data journal*, vol. 3, 2015.

[67] S. Harris and A. Seaborne, “SPARQL 1.1 Query Language W3C Recommendation,” 21 March 2013. [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[68] M. Arenas, A. Bertails, E. Prud'hommeaux and J. Sequeda, “A Direct Mapping of Relational Data to RDF W3C Recommendation,” 27 September 2012. [Online]. Available: https://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/.

[69] D. Beckett, T. Berners-Lee, E. Prud'hommeaux and G. Carothers, “Turtle Terse RDF Triple Language W3C Working Draft,” 10 July 2012. [Online]. Available: https://www.w3.org/TR/2012/WD-turtle-20120710/.

[70] F. Maali, J. Erickson and P. Archer, “Data catalog vocabulary (DCAT),” *W3c recommendation*, vol. 16, 2014.

[71] M. Dekkers, “Asset description metadata schema (adms),” *W3C Working Group*, 2013.

[72] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne and others, “OWL-S: Semantic markup for web services,” *W3C member submission*, vol. 22, 2004.

[73] H. Lausen, A. Polleres, D. Roman and others, “Web service modeling ontology (WSMO),” *W3C member submission*, vol. 3, 2005.

[74] H. Lausen, J. Bruijn, A. Polleres and D. Fensel, “WSML-a Language Framework for Semantic Web Services.,” in *Rule Languages for Interoperability*, 2005.

[75] M. Kifer, “Rule interchange format: The framework,” in *International Conference on Web Reasoning and Rule Systems*, 2008.

[76] J. Kopeckỳ, K. Gomadam and T. Vitvar, “hrests: An html microformat for describing restful web services,” in *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.

[77] J. Kopecky, T. Vitvar, D. Fensel and K. Gomadam, “hRESTS & MicroWSMO,” *CMS WG Working Draft*, p. 45, 2009.

[78] J. Kopeckỳ, T. Vitvar, C. Bournez and J. Farrell, “Sawsdl: Semantic annotations for wsdl and xml schema,” *IEEE Internet Computing,* vol. 11, pp. 60-67, 2007.

[79] M. Lanthaler and C. Gütl, “Hydra: A Vocabulary for Hypermedia-Driven Web APIs.,” *LDOW,* vol. 996, 2013.

[80] N. Mainas, E. G. Petrakis and S. Sotiriadis, “Semantically enriched open API service descriptions in the cloud,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS),* 2017.

[81] R. Hoekstra, A. Merono-Penuela, K. Dentler, A. Rijpma, R. Zijdeman and I. Zandhuis, “An ecosystem for linked humanities data,” in *European Semantic Web Conference*, 2016.

[82] G. K. (Spec-Ops), “An Extension to the Application Programming Interface for the JSON-LD Syntax,” 19 April 2019. [Online]. Available: https://w3c.github.io/json-ld-framing/.

[83] A. Meroño-Peñuela and R. Hoekstra, “grlc makes GitHub taste like linked data APIs,” in *European Semantic Web Conference,* 2016.

[84] T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing?,” *International journal of human-computer studies,* pp. 907--928, 1995.

[85] J. Farrell and H. Lausen, “Semantic annotations for WSDL and XML schema,” *W3C recommendation,* vol. 28, 2007.

[86] T. Vitvar, J. Kopecky, J. Viskova and D. Fensel, “Wsmo-lite annotations for web services,” in *European Semantic Web Conference*, 2008.

[87] R. Taelman, M. Vander Sande and R. Verborgh, “GraphQL-LD: Linked Data Querying with GraphQL,” in *ISWC2018, the 17th International Semantic Web Conference*, 2018.

[88] D. Roman, J. Kopeckỳ, T. Vitvar, J. Domingue and D. Fensel, “WSMO-Lite and hRESTS: Lightweight semantic annotations for Web services and RESTful APIs,” *Journal of Web Semantics,* vol. 31, pp. 39-58, 2015.

[89] N. Mainas, E. Petrakis and S. Sotiriadis, “Semantically enriched open API service descriptions in the Cloud,” 2017.

[90] N. Mainas, E. G. M. Petrakis and S. Sotiriadis, “Semantically enriched open API service descriptions in the cloud,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS),* 2017.

[91] I. S. A. and others, *DCAT application profile for data portals in Europe,* 2015.

[92] E. Prud, A. Seaborne and others, “Sparql query language for rdf,” 2006.