Using Semantic Technologies to Manage a Data Lake: Data Catalog, Provenance and Access Control

Henrik Dibowski[1], Stefan Schmid[1], Yulia Svetashova[1,4], Cory Henson[2], and Tuan Tran[3]

[1] Robert Bosch GmbH, Corporate Research, 71272 Renningen, Germany
henrik.dibowski@de.bosch.com
stefan.schmid@de.bosch.com
[2] Bosch Research and Technology Center, PA 15222 Pittsburgh, USA
cory.henson@us.bosch.com
[3] Robert Bosch GmbH, Chassis Systems Control, 74232 Abstatt, Germany
anhtuan.tran2@de.bosch.com
[4] Karlsruhe Institute of Technology, 76133 Karlsruhe, Germany

Abstract. Data lake architectures enable the storage and retrieval of large amounts of data across an enterprise. At Robert Bosch GmbH, we have deployed a data lake for this expressed purpose, focused on managing automotive sensor data. Simply centralizing and storing data in a data lake, however, does not magically solve critical data management challenges such as data findability, accessibility, interoperability, and re-use. In this paper, we discuss how semantic technologies can help to resolve such challenges. More specifically, we will demonstrate the use of ontologies and knowledge graphs to provide vital data lake functions including the cataloging of data, tracking provenance, access control, and of course semantic search. Of particular importance is the development of the DCPAC Ontology (Data Catalog, Provenance, and Access Control) along with its deployment and use within a large enterprise setting to manage the huge volume and variety of data generated by current and future vehicles.

Keywords: Ontology, Knowledge Graph, Semantic Data Lake, Semantic Search, Semantic Layer, Provenance, Access Control.

1 Introduction

Robert Bosch GmbH is a large enterprise company that designs and manufactures automotive components, ensuring the agility, comfort, function and safety of vehicles and driver assistance systems. Such components range from classical safety products including airbags and electronic stability control to next generation automated driving systems. Both the volume and variety of data generated by these systems have been growing dramatically in the past few years. More specifically, the types of data range from sensor data – including video, RADAR, LIDAR, and CANbus signals – to textual data and metadata about the various projects collecting and using the data within the company. To handle this complexity, we have developed a holistic architecture for managing our data within the enterprise – the Bosch Automotive Data Lake.

Simply centralizing and storing data in a data lake, however, does not immediately solve all data management challenges. Specifically, issues of findability, accessibility, interoperability, and re-use – the four principles of FAIR data[1] – remain unresolved. To facilitate these principles of FAIR data, we have extended our data lake architecture with a semantic layer. This semantic layer consists of an ontology and knowledge graph (KG) that provides meaningful, semantic description of all resources in the data lake. The resources include a heterogeneous assortment of documents, datasets, and databases. Semantic description of these resources, represented as a knowledge graph, includes information about the content of the resources, the provenance, and access control permissions. The ability to perform semantic search of all data in the data lake provides enhanced findability, access, interoperability, and re-use.

The three primary contributions of this paper include the creation of a DCPAC Ontology (Data Catalog, Provenance, and Access Control), the development of the Semantic Data Lake Catalog KG that is conformant to DCPAC, and the application of the ontology and KG for semantic search and retrieval. In Section 2, we discuss related work and then introduce the development and structure of the DCPAC Ontology in Section 3. The creation of a conformant KG and its use within an enterprise setting is explained in Section 4. Finally, in Section 5 we conclude with an overall summary and directions for the future.

2 Related Work

In the era of big data, *data catalogs* emerged as the standard for metadata management. In the last few years, however, new application areas have appeared and the volume and richness of metadata required has grown significantly. Data lakes constitute one such important new application for data catalogs, besides warehouses, master data repositories, etc. According to Gartner, a *data catalog* "… maintains an inventory of data assets through the discovery, description, and organization of *datasets*[2]. The catalog provides context to enable data analysts, data scientists, data stewards, and other data consumers to find and understand a relevant dataset for the purpose of extracting business value." [1].

Current vendors offer a wide range of commercial data catalog software. A sample of such vendors includes Alation Data Catalog, Atlan Enterprise Data Catalog, Talend Data Catalog, Collibra Data Catalog, Informatica Enterprise Data Catalog, Microsoft Azure Data Catalog, Oracle Cloud Infrastructure Data Catalog, and even Google is joining the market with its Googles Data Catalog. To our knowledge, however, none of these data catalogs uses or supports standard semantic technologies, nor do they allow for using existing ontology vocabularies. Rather they are closed, propriety systems with their own metadata languages and glossaries.

[1] https://www.go-fair.org/fair-principles/

[2] *Datasets* are the files, tables, graphs etc. that applications or data engineers need to find and access.

Anzo Cambridge Semantics[3] is one of a few exceptions, as it is built from the open data standards OWL, RDF and SPARQL, which makes it simple to leverage rapidly evolving vocabularies in multiple industries. Anzo has a built-in smart data catalog functionality that is able to automatically extract the schemas of databases in a data lake and support the mapping of the schemas to ontology terms. But the integration of this data catalog functionality with existing ETL pipelines, as well as extensibility of the built-in data catalog ontology based on domain specific needs, is limited.

Adding a *semantic layer* to a data lake is a common approach to developing a *semantic data lake*, which have been described in literature. The use of data catalogs in this context, however, are still rare. In [2], a data lake using semantic technologies is presented that can manage datasets produced by sensors or simulation programs in the manufacturing domain. It comprises a data catalog that provides inventory services and also implements security mechanisms. Different from our approach, however, this data catalog is not built using standard semantic technologies, but rather as a simple file system.

A *semantic data lake architecture* for autonomous fault management in software-defined networking environments, with clear similarities to ours (Section 4), is described in [3]. Another comparable semantic data lake architecture called “Squerall” is proposed in [4]. This solution proposes distributed query execution techniques and strategies for querying heterogeneous big data. Both approaches, however, lack a data catalog and other means of handling provenance or access control.

Our solution differs from existing solutions by proposing a semantic data lake architecture that incorporates a semantic data catalog, built with standard semantic technologies, and that addresses provenance and access control for resources in the data lake. This solution is described in detail in the following sections.

3 Semantic Data Catalog, Provenance and Access Control Layer for Data Lakes

As one of the three primary contributions of this paper, this section describes the DCPAC ontology (Data Catalog, Provenance, and Access Control). The DCPAC ontology can be applied for adding a semantic layer to a data lake, which provides semantic description of the content, provenance, and access control permissions of the resources in a data lake. This ontology was created by combining several common, (predominantly) standardized ontology vocabularies and by aligning and extending them where necessary.

3.1 Ontology Layer Architecture

Fig. 1 shows a layer architecture diagram of DCPAC ontology, including the ontology vocabularies used and their import-relationships. The DCPAC ontology is shown at

[3] https://www.cambridgesemantics.com/product/data-cataloging/

the bottom, and recursively imports all other ontologies. Additionally, it defines SHACL constraints for validating instance data (ABox).

In the following subsections, the primary ontologies utilized by DCPAC are described.

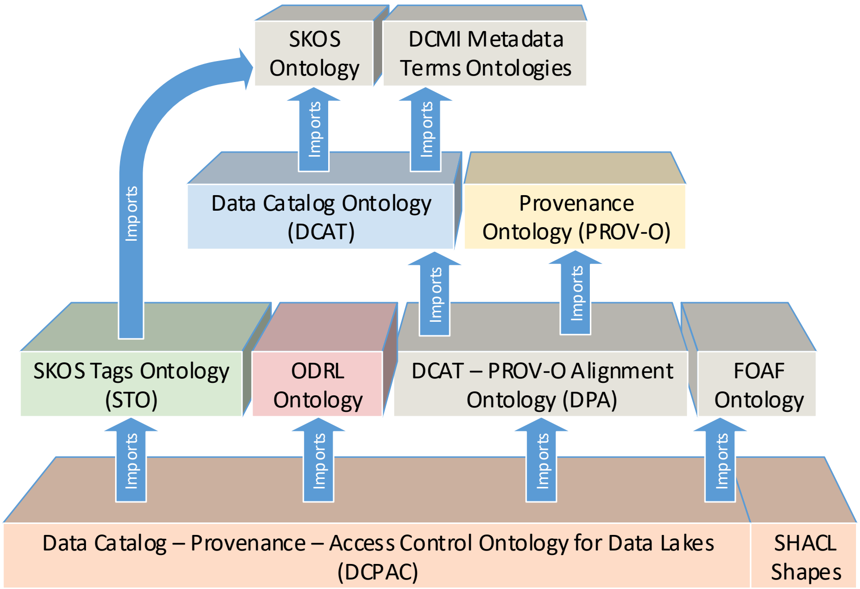

SKOS Ontology
DCMI Metadata Terms Ontologies
Imports
Data Catalog Ontology (DCAT)
Provenance Ontology (PROV-O)
SKOS Tags Ontology (STO)
ODRL Ontology
DCAT – PROV-O Alignment Ontology (DPA)
FOAF Ontology
Data Catalog – Provenance – Access Control Ontology for Data Lakes (DCPAC)
SHACL Shapes

Fig. 1. Layer architecture of the data catalog, provenance and access control (DCPAC) ontology for data lakes.

Data Catalog (DCAT) Ontology [Prefix: dcat]. The Data Catalog (DCAT) ontology "… is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. … DCAT enables a publisher to describe datasets and data services in a catalog using a standard model and vocabulary that facilitates the consumption and aggregation of metadata from multiple catalogs. This can increase the discoverability of datasets and data services. It also makes it possible to have a decentralized approach to publishing data catalogs and makes federated search for datasets across catalogs in multiple sites possible using the same query mechanism and structure." [5]. DCAT is standardized as a W3C recommendation, with the latest version from February 2020, and is being developed further by an active community.

The DCAT ontology imports and uses the widely recognized SKOS [6] and DCMI Metadata Terms [7] ontologies. Its primary purpose in the context of the DCPAC ontology is the semantic description of the content of resources in a data lake.

Provenance Ontology (PROV-O) [Prefix: prov]. The Provenance Ontology (PROV-O) "… expresses the PROV Data Model using the OWL2 Web Ontology Language. It provides a set of classes, properties, and restrictions that can be used to represent and interchange provenance information generated in different systems and under different contexts. It can also be specialized to create new classes and properties to model provenance information for different applications and domains." [8]. PROV-O is a W3C recommendation from April 2013. Its purpose in the context of DCPAC is to describe the provenance of the data lake resources. Such provenance information may include the ownership of resources how they were created, by which activity and agent, and from what data they were derived.

Open Digital Rights Language (ODRL) Ontology [Prefix: odrl]. The Open Digital Rights Language (ODRL) ontology "… is a policy expression language that provides a flexible and interoperable information model, vocabulary, and encoding mechanisms for representing statements about the usage of content and services. The ODRL Vocabulary and Expression describes the terms used in ODRL policies and how to encode them." [9]. The latest version 2.2 was published by the W3C in September 2017. In our data lake scenario, ODRL is applied to defining access control permissions for the data lake resources, including who can access a resource and which actions are permitted, i.e. display, read, modify, delete.

DCAT – PROV-O Alignment (DPA) Ontology [Prefix: dpa]. The DCAT – PROV-O Alignment (DPA) ontology [10] was created by the W3C Dataset Exchange Working Group (DXWG) and contains alignment axioms between DCAT ontology and PROV-O. Thereby, it enhances the DCAT ontology with the ability to use PROV-O for expressing advanced provenance information.

The most relevant alignments defined in the DPA ontology are shown in Fig. 2. It aligns the DCAT ontology classes `dcat:CatalogRecord`, `dcat:Resource` and `dcat:Distribution` as subclasses of the PROV-O class `prov:Entity` by adding corresponding `rdfs:subClassOf` statements. Thus, all instances of these classes and their subclasses become instances of `prov:Entity`, which allows the usage of all associated PROV-O object properties and classes for modeling provenance information. This makes the provenance and authorship of data, along with its evolution over time, trackable in each little detail.

SKOS Tags Ontology (STO). The Simple Knowledge Organization System (SKOS) is "a common data model for sharing and linking knowledge organization systems" [6]. We design separate SKOS vocabularies for different domains and use them to specify the semantics of resources in a data lake, dependent on its subject. In particular, we assign each dataset a set of `skos:Concept`s as tags that provide semantic description of the content of a data lake resource.

The SKOS vocabularies are domain specific. While defining these vocabularies, we often reuse terms from existing or newly developed domain ontologies. From the domain ontologies, we select subsets of classes and individuals that are relevant for the tasks of retrieval, and define them as instances of `skos:Concept`. Domain specific SKOS vocabularies are iteratively added to the SKOS Tags ontology (STO), which serves as a generic component of the domain-agnostic architecture of the

DCPAC ontology (see Fig. 1), bridging it with domain specific ontologies. In the following sub-section, we describe one such domain ontology (ASO), developed for the Bosch Automotive Data Lake, and show its relationship to the STO.

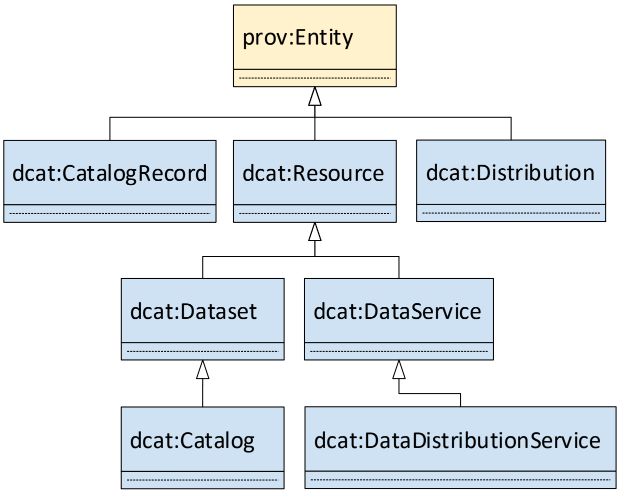


Fig. 2. DPA ontology: Alignment of DCAT ontology with PROV-O.

Automotive Signal Ontology (ASO) [Prefix: aso]. The primary goal of the Automotive Signal Ontology [11] is to represent manifold signal types in automotive datasets and to enable non-trivial queries spanning over datasets of different types, formats and modalities (including radar signals, onboard diagnostics and video data). The use of this ontology allows non-domain experts to understand and query the data, as well as to automate the integration of signals from different sources in support of a wide range of applications and use-cases of interest to the automotive industry.

The ASO is an OWL 2 ontology. It borrows concepts from several standard ontologies and vocabularies, namely the W3C Semantic Sensor Network Ontology (SSN) [12], the Quantities, Units, Dimensions, and Data Types Ontologies (QUDT) [13] and the Vehicle Signal and Attribute Ontology (VSSo) [14], generated from the automotive standard VSS [15]. The ASO conceptualizes a signal by defining several meaningful relations, including the signal type (e.g. aso:WindowPosition as a subclass of aso:ObservableSignal), the associated vehicle component (e.g. aso:Window), the sensor(s) and actuator(s) involved in generating signal data, as well as the measured physical quantities and units-of-measure. It also provides terms to describe the specific details of automotive data collection, e.g. CAN bus data, CAN frames, messages and signals.

The ASO also defines an associated SKOS vocabulary, where all signals are defined as instances of skos:Concepts. This vocabulary is a part of STO.

Consequently, the ASO has a dual role in our Automotive Data Lake. The typing of ASO signals as skos:Concepts provides the means to tag resources in the data lake in a consistent way and enriches the semantic search capabilities provided by our

DCPAC ontology. In addition, the formal semantics of the ASO itself enables expressive queries, which go beyond the hierarchical SKOS tag search and make the data lake truly semantic. For example, find all datasets that are tagged with signals of a certain type (e.g. `aso:ObservableSignal`) and being associated with specific vehicle component (e.g. `aso:Window`).

3.2 Data Catalog – Provenance – Access Control (DCPAC) Ontology [Prefix: dcpac]

The DCPAC ontology is our primary contribution to the ontology layer architecture shown in Fig. 1. It combines, aligns and extends the ontology vocabularies described in the previous section. The ontology directly imports the ODRL ontology, the DPA ontology, the FOAF ("Friend of a Friend") ontology [16] and optionally one or more STO ontologies, and recursively imports all other shown ontologies. We chose to reuse properties defined by the FOAF ontology – such as `foaf:givenName`, `foaf:name`, and `foaf:mbox` – to extend the existing definitions of `prov:Agent` and `odrl:PartyCollection`.

Alignments and Extensions to the Upper Layer Ontologies. The DCPAC ontology aligns the DCAT ontology with the ODRL ontology by declaring the classes `dcat:Distribution` and `dcat:Resource` to be subclasses of `odrl:Asset`, as can be seen in the upper part of Fig. 3. With `odrl:Asset` representing a resource or a collection of resources that are the subject of access authorization rules, this enables the definition of access control permissions for these DCAT classes and subclasses with the ODRL vocabulary. Furthermore, the DCPAC ontology extends the DCAT ontology by defining various types of `dcat:Dataset` subclasses (see Fig. 3), which allows for distinguishing different types of datasets in a data lake, such as raw data files, tabular data files, relational database and graph database resources.

Another contribution is the alignment of PROV-O with the ODRL ontology, as shown in Fig. 4. The PROV-O class `prov:Agent` is declared as subclass of `odrl:Party`, hence enabling all instances of `prov:Agent` to undertake roles in access control permissions. Additionally, the DCPAC ontology defines new subclasses of `prov:Activity`, which allow for distinguishing different types of activities that created (`dcpac:GenerationActivity`) or modified (`dcpac: ModificationActivity`) a data lake resource.

SHACL Constraints. The DCPAC ontology is associated with a SHACL shapes definition file that defines a comprehensive set of *SHACL constraints* of type SHACL Shapes (Node Shapes, Property Shapes) and *SPARQL-based constraints* [17]. SHACL shapes define cardinalities and type restrictions on properties, and regular expressions on the allowed values of string datatype properties. One such SHACL shape, for example, validates that each `dcat:Dataset` instance has to have exactly one value of type string defined for the property `dct:identifier`, and the string must match the regular expression "`^[a-z0-9][a-z0-9_\-]{2,59}$`". SPARQL-based constraints have a higher expressivity and can capture complex dependencies as graph patterns. For the class `dcat:Dataset`, for example, we defined a constraint that validates that each instance must have at least one semantic tag (`skos:Concept`) attached, and the tags must be members of a `skos:ConceptScheme` that is associated with (i.e. enabled for) the catalog the dataset belongs to (see also next section and Fig. 5).

A SHACL engine can process the constraints and validate the consistency of the KG (ABox)[4]. That improves the integrity and quality of the KG and prevents issues.

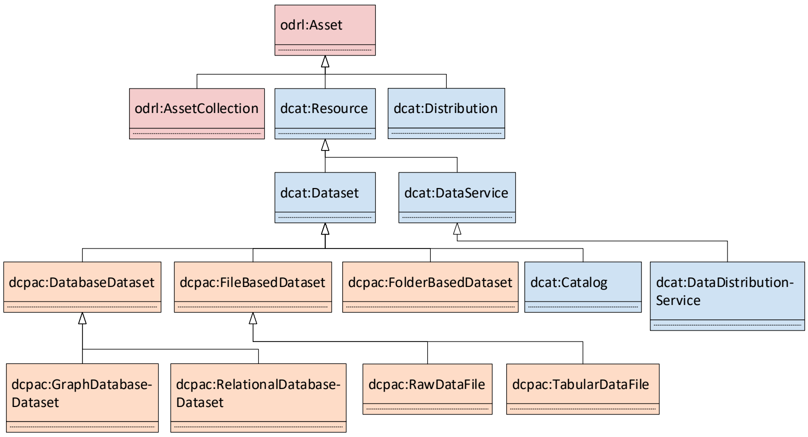


Fig. 3. DCPAC ontology: Refinement and alignment of DCAT ontology with ODRL ontology.

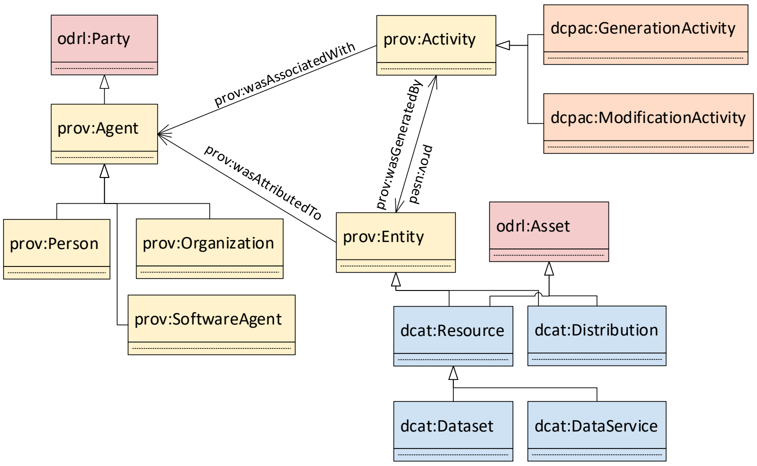


Fig. 4. DCPAC ontology: Refinement and alignment of PROV-O with ODRL ontology.

[4] We use Stardog as highly scalable triple store for our Bosch data lake. Stardog supports SHACL and has an inbuilt SHACL engine. https://www.stardog.com/platform/

3.3 The Core Vocabulary.

This Section provides an overview and explanation of the core vocabulary of the DCPAC ontology and the primary imported vocabularies, which are explained in the previous sections. For the explanation, we refer to Fig. 5, which shows the main ontology classes as well as the most important object properties and datatype properties. The stereotypes shown for some of the classes in Fig. 5 contain their superclasses and hence their alignment to the other ontologies described in the previous sections. We abstain from showing and explaining additional classes and properties that are specific for the Bosch Automotive Data Lake in order to maintain comprehensibility and domain-independence.

DCAT Entities. Let us start with the DCAT ontology classes shown in the center and bottom left of Fig. 5. The overall data catalog of the data lake is represented by one instance of class `dcat:Catalog`. It can contain many `dcat:Dataset` instances, one per resource in the data lake, e.g. raw data files, HBase or Hive tables, or RDF-based knowledge graphs. An instance of class `dcat:Distribution` models a specific representation of a dataset, comprising a specific serialization or schematic arrangement. Different distributions can exist for the same dataset, and are accessible via a URL (`dcat:downloadURL`). The data catalog and the datasets can each have several data distribution services (`dcat:DataDistributionService`), which are end-points that provide access. They are accessible via an endpoint URL (`dcat:endpointURL`).

PROV-O Entities. The PROV-O classes and properties shown in the top right part of Fig. 5 are used for modeling the provenance of the data catalog and its datasets (both declared as subclasses of `prov:Entity`, see Fig. 2), and for defining agents (e.g. person, software agent) they are attributed to (`prov:wasAttributedTo`) or that were involved in the activity of creating the dataset. Activities (`prov:Activity`) are initiated by agents (`prov:wasAssociatedWith`), create new Datasets (`prov:wasGeneratedBy`), have an start and end time, and can use other datasets as input (`prov:used`).

ODRL Entities. Access control is defined by classes and properties from the ODRL ontology. An `odrl:Permission` can define an access rule for groups of agents (`odrl:PartyCollection`) to datasets, their distributions and/or data distribution services (`odrl:target`). The allowed actions (`odrl:Action`), such as display, read, modify, delete, are defined as `skos:Concept` and attached via `odrl:action` object property.

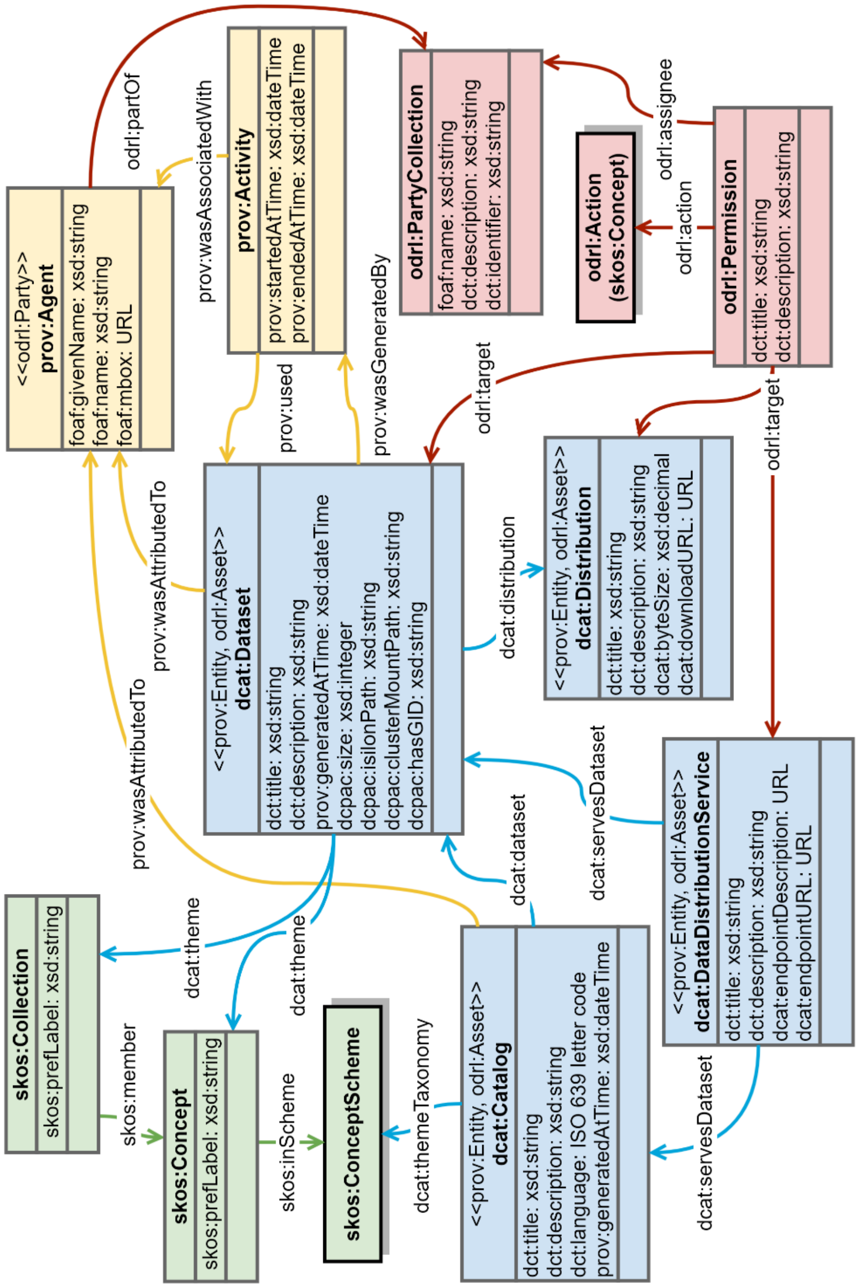


Fig. 5. The main classes and properties of the DCPAC ontology (TBox).

SKOS Entities. SKOS finally is applied for defining the semantics of the content of a dataset. Therefore, the catalog refers to one or more sets of SKOS concepts (`skos:ConceptScheme`) that can be used for semantically tagging datasets. The defined SKOS tags can be either directly linked to a dataset (`dcat:theme`), or they can be bundled and linked as a collection (`skos:Collection`), which enables the definition and reuse of (large) sets of SKOS tags. For the Bosch Automotive Data Lake we use the ASO ontology, as described in Section 3.1.

4 Semantic Data Lake Catalog

At Bosch, we have built an Automotive Data Lake as a centralized platform for the engineering and testing of our autonomous driving applications [11]. To handle and manage the complexity and enormous volume of data from all our test drives, we have developed a holistic architecture, which is shown in Fig. 6. and explained in this section. Resources collected and stored in the Bosch Automotive Data Lake include a heterogeneous assortment of documents, datasets and databases. We have created a semantic layer, the **Semantic Data Lake Catalog**, which provides meaningful semantic description of resources in the data lake and enables semantic search. The Semantic Data Lake Catalog comprises a knowledge graph that is built with the vocabulary defined in the DCPAC ontology (see Section 3). The semantic description of the resources includes information about their content, provenance and access control permissions. The ability to perform semantic search of all data in the data lake provides enhanced findability, access, interoperability, and re-use.

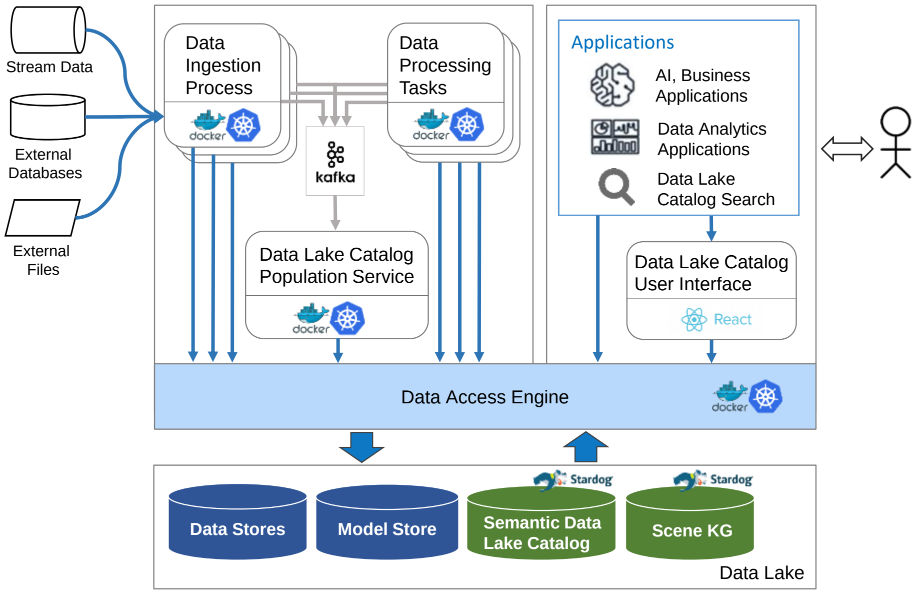


Fig. 6. Data lake architecture and role of Semantic Data Lake Catalog.

In the sub-sections below, we explain the other components of the Automotive Data Lake shown in Fig. 6, and clarify the process by which the Semantic Data Lake Catalog KG is populated and how it is used to query, find and access data assets.

Data Ingestion Process. As illustrated in Fig. 6, external data from different sources (test fleet vehicles, test benches, data warehouse, etc.) are ingested into the data lake, either continuously in streams or driven by users and applications. The ingestion process is responsible for extracting, transforming and loading new data assets into the data stores. The implementation of this ingestion process was containerized using our in-house DevOps tool in order to allow scaling-out based on the load. It is important to note that this tool does not only provide a mechanism to deploy the ingestion processes on our on-premises infrastructure, it also wraps the ingestion with a list of standard operators that are automatically called to report the process information as well as input & output data to a Kafka[5] cluster. These reports, published as standardized Kafka messages, are consumed by the Data Lake Catalog Population Service.

Data Lake Catalog Population Service. Triggered by Kafka messages, the Data Lake Catalog Population Service reads the available metadata on the ingested data assets and constructs the relevant semantic data as input for our Semantic Data Lake Catalog. The Data Lake Catalog Population Service aligns, annotates and enriches the input data from the Data Ingestion Process with DCPAC concepts before populating the Semantic Data Lake Catalog[6]. Besides dictionary based mappings (i.e. input data schema terms or tags are mapped to dedicated SKOS concepts of our Semantic Data Lake Catalog taxonomies), the population service also links signal name strings to relevant automotive signal concepts from the Automotive Signal Ontology (based on Levenshtein distance). This is a critical part of the knowledge construction process, as it enables us to search, integrate and process the various data assets based on a shared conceptualization. The Data Lake Catalog Population Service will also record relevant provenance information; e.g. the activity that has created or modified a data asset, including information about the source asset as well as begin and end time.

Data Access Engine. The Data Access Engine (DAE) provides applications with a uniform query interface and access to resources (e.g. files, tables, knowledge graphs) in the data lake based on a common HTTP-based API and endpoint. At this stage, the DAE supports data-type queries (i.e. in HBase[7] or Hive[8]/Impala[9] tables), knowledge queries (i.e. SPARQL queries of knowledge graphs) and task requests (i.e. Oozie[10] jobs in the Hadoop[11] cluster). The DAE secures and hides the details of the underlying

[5] Apache Kafka: A distributed streaming platform, https://kafka.apache.org/

[6] We use Stardog for storing and processing the semantic layer as knowledge graph.

[7] Apache HBase: Distributed big data store that runs on Hadoop, https://hbase.apache.org/

[8] Apache Hive: Data warehouse software for large distributed datasets, https://hive.apache.org/

[9] Apache Impala: Native analytic database for Apache Hadoop, https://impala.apache.org/

[10] Apache Oozie: Workflow scheduler for Hadoop, https://oozie.apache.org/

[11] Apache Hadoop: Scalable, distributed computing software, https://hadoop.apache.org/

storage system and enables transparent re-direction of requests based on stable and global identifiers. The DAE uses the Semantic Data Lake Catalog to control access to individual data assets based on common access operations (e.g. read, modify, delete). In particular, the Semantic Data Lake Catalog provides a list of authorized user groups for a given dataset according to its content type, security class and assigned project. For authorization, the DAE supports Kerberos[12], for easy integration with Enterprise IT systems, as well as JSON Web Token[13] based authorization. Besides the access control, the DAE also supports template-based requests for the different types of resources. The DAE fetches the respective template from the Semantic Data Lake Catalog and fills the template with parameters provided in the request. Once the template is complete, the DAE fetches/queries the respective resources in the data lake and returns the results to the client. This template based request feature has proven especially useful in the case of knowledge graph queries, as it enables frontend developers – without knowledge of RDF and SPARQL – to query relevant data for their applications.

Besides using templates, administrators and authorized data engineers are also able to query the data lake and Semantic Data Lake Catalog with native SPARQL queries via the DAE. This enables privileged users to carry out complex semantic search and analytics on the Semantic Data Lake Catalog (e.g. find all data assets that have been derived from a given asset, or find all datasets that contain signals associated with a particular sub-system of a car), or perform advanced data management operations (e.g. delete all data assets computed by a given provenance activity or task). The DAE also provides a means for data management agents to query and update the Semantic Data Lake Catalog programmatically. This is the basis for implementing sophisticated data lake management agents that can leverage the full semantic query capabilities of our Semantic Data Lake Catalog.

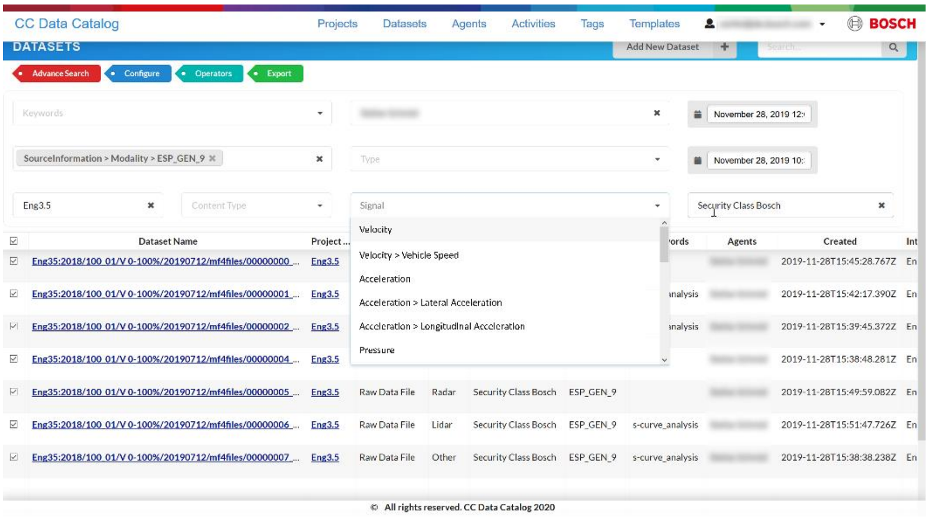
Data Processing Tasks. Similar to the Data Ingestion Process, our run-time environment also supports containerized data processing tasks, such as data enrichment, knowledge construction, and data analytics. These tasks use the Semantic Data Lake Catalog to find data resources and the DAE to access data or knowledge in the data lake. Such processing tasks typically create new data resources (e.g. knowledge graphs or tables) or curate existing ones to persist the results. Whenever new data resources are created or further metadata about resources are discovered, the Semantic Data Lake Catalog is automatically updated. The Data Lake Catalog Population Service is triggered via corresponding Kafka messages. Consequently, the Semantic Data Lake Catalog is automatically updated and relevant provenance information are recorded.

Data Lake Catalog User Interface. For data lake administrators and data engineers to manage and search data resources available in the data lake, we developed a Web application that allows the search and selection of resources based on the available

[12] Kerberos: The network authentication protocol, http://web.mit.edu/kerberos/

[13] JSON Web Token, https://tools.ietf.org/html/rfc7519

meta data (e.g., type of resource, content type, signals, recording date, associated SKOS tags). Selected data resources can then be batch processed, e.g. curated with relevant SKOS tags or keywords, ownership or permissions changed, or deleted from the data lake. Fig. 7 illustrates a screenshot of our Data Lake Catalog Web application.



[redacted]

SourceInformation > Modality > ESP_GEN_9

Eng3.5

Security Class Bosch

November 28, 2019 12:

November 28, 2019 10:

☑	Dataset Name	Project ...					ords	Agents	Created	Int
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000000_...	Eng3.5						[redacted]	2019-11-28T15:45:28.767Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000001_...	Eng3.5					nalysis	[redacted]	2019-11-28T15:42:17.390Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000002_...	Eng3.5					nalysis	[redacted]	2019-11-28T15:39:45.372Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000004_...	Eng3.5						[redacted]	2019-11-28T15:38:48.281Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000005_...	Eng3.5	Raw Data File	Radar	Security Class Bosch	ESP_GEN_9		[redacted]	2019-11-28T15:49:59.082Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000006_...	Eng3.5	Raw Data File	Lidar	Security Class Bosch	ESP_GEN_9	s-curve_analysis	[redacted]	2019-11-28T15:51:47.726Z	En
☑	Eng35:2018/100_01/V 0-100%/20190712/mf4files/00000007_...	Eng3.5	Raw Data File	Other	Security Class Bosch	ESP_GEN_9	s-curve_analysis	[redacted]	2019-11-28T15:38:38.238Z	En

© All rights reserved. CC Data Catalog 2020

Fig. 7. A screenshot of the Data Lake Catalog User Interface.

Our Semantic Data Lake Catalog has been in use since the end of 2019. The Data Lake Catalog Population Service is now continuously populating the Semantic Data Lake Catalog as new data resources are ingested to the data lake. Since the data lake was already in use prior to the implementation of the automated population process described above, we are still batch-processing data ingested in the past based on various metadata sources.

Table 1 shows the number of data resources registered in the Semantic Data Lake Catalog, the number of facts in the knowledge graph, the data volume of the registered resources as well as the number of registered users.

Table 1. Statistics of the Semantic Data Lake Catalog KG and data lake (as of 2020-03-31).

Types of Data Resources	6
Data Resources	1,798,927
Facts (triples) in KG	32,363,911
Fact to Data Resource ratio	~18:1
RDF Store Cluster Size	3 Stardog Servers
Data Volume (all assets)	64.1648 Petabytes
Projects	10

5 Conclusion

This paper presents the design and implementation of a semantic layer for data lakes in general and reports on its realization and use for managing data in the Bosch Automotive Data Lake. In particular, we demonstrate how we use our DCPAC ontology (Data Catalog, Provenance, Access Control) for managing data resources in a comprehensive manner, enabling findability, accessibility, interoperability, and re-use – the four principles of FAIR data (Section 3). We report on the concrete application of the DCPAC ontology in conjunction with our Automotive Signal Ontology (ASO) in the implementation of the Semantic Data Lake Catalog at Bosch (Section 4). At the core of our Semantic Data Lake Catalog is a knowledge graph that includes instances for all data resources in our data lake (~1.8M) comprising of the available metadata (id, name, date created, date modified, size, format, etc.). The knowledge graph also stores references to the encompassing automotive signals (ASO) being defined as SKOS tags, defines access control permissions, and documents provenance information such as activities and associated agents. This comprehensive knowledge graph offers our data scientists and engineers sophisticated semantic search and data management capabilities, by combining typical metadata search with semantic search based on content-related (SKOS tags and formal semantics of the ASO) as well as provenance-related (entities, activities, agents) information.

Several important lessons learnt from the design and usage of the system so far include: (1) The DCPAC ontology in conjunction with ASO has proven sufficiently expressive to find and manage the automotive data resources in our Semantic Data Lake Catalog; (2) The population of the Data Lake Catalog must be fully automatic and triggered by the data ingestion process; (3) Semantic search and management of data resources based on SKOS tags of automotive signals and their conceptualizations are critical as equivalent signals occur with different names across the enterprise; (4) Provenance related information are critical to manage data and enable traceability and automatic reprocessing/updating of derived data; (5) Access control can be seamlessly integrated into the KG.

One of the limitations of our current system is that the ASO covers only a small fraction of the overall signals used in our data. This is because the ASO has been manually populated so far. As future work, we target a processing pipeline that populates the ASO with missing signals in a semi-automatic manner by involving domain experts as needed.

References

1. Edjlali, R., Duncan, A. D., De Simoni, G., Zaidi, E.: Data Catalogs Are the New Black in Data Management and Analytics. Gartner Research (2017).
2. Kasrin, N., Qureshi, M., Steuer, S., Nicklas, D.: Semantic Data Management for Experimental Manufacturing Technologies. Datenbank-Spektrum 18(1), 27-37 (2018).
3. Benayas, F., Carrera, A., Amado, M. G., Iglesias, C. A.: A semantic data lake framework for autonomous fault management in SDN environments. Transactions on Emerging Tele-

communications Technologies, Special Issue on Machine Learning/AI for IoT, M2M, and Computer Communication 30(9) (2019).

4. Mami, M. N., Graux, D., Scerri, S., Jabeen, H., Auer, S., Lehmann, J.: Uniform Access to Multiform Data Lakes using Semantic Technologies. In: Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services (iiWAS2019), pp. 313-322, Munich, Germany (2019).
5. Albertoni, R., Browning, D., Cox, S., Beltran, A. G., Perego, A., Winstanley, P.: Data Catalog Vocabulary (DCAT) – Version 2. W3C Recommendation (2020), https://www.w3.org/TR/vocab-dcat-2/, last accessed 2020/03/26.
6. Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System Reference. W3C Recommendation (2009), https://www.w3.org/TR/skos-reference/, last accessed 2020/03/31.
7. Dublin Core Metadata Initiative Board: DCMI Metadata Terms. DCMI Recommendation (2020), https://www.dublincore.org/specifications/dublin-core/dcmi-terms/, last accessed 2020/03/26.
8. Lebo, T., Sahoo, S., McGuinmess, D.: PROV-O: The PROV ontology. W3C Recommendation (2013), http://www.w3.org/TR/prov-o/, last accessed 2020/03/12.
9. Iannella, R., et. al.: ODRL Version 2.2 Ontology. W3C (2017), https://www.w3.org/ns/odrl/2/, last accessed 2020/03/26.
10. W3C Dataset Exchange Working Group (DXWG): DCAT-PROV alignment ontology. W3C (2019), https://github.com/w3c/dxwg/blob/gh-pages/dcat/rdf/dcat-prov.ttl, last accessed 2020/03/26.
11. Schmid, S., Henson, C., Tran T.: Using Knowledge Graphs to Search an Enterprise Data Lake. In: Hitzler, P. et al. (eds) The Semantic Web: ESWC 2019 Satellite Events. ESWC 2019. Lecture Notes in Computer Science, vol. 11762, pp. 262-266. Springer, Cham (2019).
12. Haller, A., et. al.: The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web 10(1), 9–32 (2019).
13. Hodgson, R., Keller, P. J., Hodges, J., Spivak, J.: QUDT: quantities, units, dimensions and data types ontologies. QUDT.org (2014), http://qudt.org/ (2014), last accessed 2020/03/31.
14. Klotz, B., Troncy, R., Wilms, D., Bonnet, C.: VSSo: A Vehicle Signal and Attribute Ontology. In: Lefrançois, M., García-Castro, R., Gyrard, A., Taylor, K. (eds.) Proceedings of the 9th International Semantic Sensor Networks Workshop co-located with 17th International Semantic Web Conference (SSN@ISWC), CEUR Workshop Proceedings, vol. 2213, pp. 56–63 (2018).
15. GENIVI Alliance: Vehicle Signal Specification. (2017), https://github.com/GENIVI/vehicle_signal_specification, last accessed 2020/09/21.
16. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.99. (2014), http://xmlns.com/foaf/spec/, last accessed 2020/03/26.
17. Kublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation (2017), https://www.w3.org/TR/shacl/, last accessed 2020/03/26.