Design and Implementation of an Ontology for Semantic Labeling and Testing: Automotive Global Ontology (AGO)

Itziar Urbieta a, Marcos Nieto a, Mikel García a and Oihana Otaegui a

a Intelligent Transportation Systems and Engineering, Vicomtech, Mikeletegi Pasealekua 57, Spain
E-mails: iurbieta@vicomtech.org, mnieto@vicomtech.org, mgarcia@vicomtech.org, ootaegui@vicomtech.org

Abstract. Modern Artificial Intelligence (AI) methods are able to massively produce accurate and rich descriptions of data, in domains like surveillance or automation. The need to organize data at scale in a semantic structure has then arisen for long-term data maintenance and consumption. Ontologies and graph databases have gained popularity as a mechanism to satisfy this need. Ontologies provide the means to formally structure descriptive and semantic relations of a domain. Graph databases allow efficient and well-adapted store, manipulation and consumption of these linked-data resources. However, up to date, there is not a universally defined strategy for building AI-oriented ontologies for the automotive sector. One of the key challenges is the lack of a world-wide standardised vocabulary. Most private initiatives and large open datasets for Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) development include their own definition of terms, with incompatible taxonomies and structures producing a well-known lack of interoperability. This paper presents a methodology for designing and building domain ontologies as a Knowledge Organization System (KOS) using graph databases (Neo4j). The Automotive Global Ontology (AGO) is issued as well as a result of the methodology implementation. Two different use cases for AGO are presented to showcase its capabilities: semantic labeling and scenario-based testing. The ontology and related material is made public for its subsequent usage by the industry and academic communities.

Keywords: Semantics, ontology, scenario-based testing, ADAS, graph database, Neo4j.

1. Introduction

Labeling, in the context of Artificial Intelligence (AI), is the process of adding descriptive information to data, for instance, collections of images, data series, or sensor measurements. Labeling has been recently identified as one of the major bottlenecks for machine learning (ML) progress, as current state of the art on deep learning (DL) implies creating massive training datasets composed of data samples (e.g. images, point cloud scans, etc.) labeled with the description the DL model will learn (e.g. labels with the class name, or the bounding box or shape of objects). As a consequence, better models are typically obtained using larger and carefully crafted datasets, which cover as much as possible all the potential data variability of the domain of interest (e.g. all possible configuration of complex driving scenes).

Labeling business models have emerged consequently, where label producers offer services to create large datasets out of raw data recordings. As an example, from 2018 to 2020, almost 20 very large open datasets have been released for the development of autonomous driving (AD) technologies (see nuScenes1, Lyft Level 52, H3D Honda Dataset3, Waymo4, Audi A2-D25, Berkeley Deep-Dive (BDD)6, Apolloscape7, or Mapillary Vistas8). Such approaches solve the short-
term need for labels on DL training stages, but open
new challenges for long-term operation: evolution of
label types, class taxonomies and hierarchies; fusion
or comparison of labels from heterogeneous sources;
scene management and searching via complex queries,
etc. Other problems are: usage of human languages
and synonyms, different taxonomies (hierarchies of
classes), different types of attributes and properties,
etc. This in turn leads to the problem of having to de-
velop a new model for each dataset. Hence, dataset
labeling approaches present the requirement of repre-
senting data linked to ontology-based semantics [17].

Analogously, scenario-based testing requires the
ability to unambiguously describe scenes from an im-
perative perspective, i.e. to command its generation or
implementation in testing methods such as simulation
environments or field testing. Scenario representations
need to refer to terms that correspond to classes de-

dined in a KOS to guarantee simulation repeatability,
and results traceability. Large databases of scenarios
need to be explored via queries, to draw from them
particular data subsets of interest for their consump-
tion for testing specific AD functions. Semantic con-

nection between elements of a scenario description can
lead to semantic querying, i.e. the ability to find hidden
or non-explicit information using rules or reasoning-
enabled query mechanisms.

Furthermore, data science has gained importance in
the last years since the quantity of collected data is in-
creasing every day. Data has evolved to become crucial
for strategic decision making or situation-awareness
based systems. Likewise, data-driven DL approaches
are emerging in the automotive industry. These de-
velopments require higher-level semantics to enhance
the performance, functionality and scope of trained
models, enabling reasoning mechanisms towards ac-
tion recognition or scene understanding.

In this paper we present our approach towards
the construction of the Automotive Global Ontology
(AGO) using graph databases to enable semantic ser-

vices for the automotive domain. The novelty of our
approach is the proposed methodology to build such
ontology from diverse, heterogeneous data sources and
taxonomies (e.g. from large existing Advanced Driver
Assistance Systems (ADAS) and Autonomous Driving
(AD) open datasets). The proposed method ensures a
global list of concepts is created, linked to the exist-
ing source concepts but at the same time that general-
izes well for future datasets. AGO covers the need
to provide meaning to labels and scenario descriptions
by including high-level semantics in the knowledge
base. Thus, the format of data is detached from mean-
ing, and standard practices can be safely used, such as
the Video Content Description (VCD)9 language or the
upcoming ASAM OpenLabel10 standard. The
VCD structure permits labeling an entire scene in a
single file including actions, attributes and relations in
the form of RDF triples [18] which makes it ideal for
the considered use cases.

The paper is organized as follows: existing semantic
resources in the field are studied in section 2; section 3
presents the terminology defined for this work, while
section 4 introduces the AGO domain model. Section 5
details the methodology and the ontology construction
process. Finally, section 6 exemplifies AGO utilisation
on the two defined use cases, with the produced results
presented in section 7.

2. State of the art

Different works in the transportation field have iden-
tified the importance of domain-knowledge structures
for different purposes: to assess traffic scenes in real
time applications [11], to provide automatic support
for design and analysis of performance monitoring
systems for Public Transport Systems [10], or to infer
knowledge to aid test management [21].

Several ontologies in the transportation domain can
be found in the literature, e.g. Ontology of Traffic Net-

works (OTN) [4], which is summarized as a direct en-
coding of Geographic data files (GDF) in OWL, and
Transport Disruption ontology [7] that provides a for-

mal framework for modeling travel planning-related
events.

In recent years, the focus has been set into knowledge-
based approaches representing scenarios with the pur-
pose of promoting the scenario-based evaluation of
ADAS and AD. Ontologies have become a key compo-

ten for formalizing this knowledge. An event-based
scenario description for testing was presented [29]

based on the three abstraction levels for scenario de-

scription: functional, logical and concrete scenarios, as
described in [28] and adopted by the Pegasus11 project.
The strategy was enhanced with a procedure centered
in allowing for qualitative description to generate more
concrete scenarios. This scenario-based testing strat-
	
ey entails the development of a structured knowl-

9http://vcd.vicomtech.org
10https://www.asam.net/active-projects/
11https://www.pegasusprojekt.de/en/
edge and formal scenario representation language for driving simulation environments, such as, OpenSCENARIO\textsuperscript{12}, CommonRoad\textsuperscript{13} or the Safety Pool’s scenario description language\textsuperscript{14}.

Despite these advances, currently an open knowledge-base of the Automotive domain that covers the needs of the testing and labeling applications is missing. Therefore, one of the aims of this paper is to present AGO. This has been built with the purpose of formalizing the terminology used for representing automotive scenarios and providing the required knowledge layer to support semantic-labeling tools.

A few works have been published related to ontology engineering methodologies that present design and construction principles. However, to the best of our knowledge, there is not yet a standardized approach or any formal requirement other than the ontology languages defined by the W3C group \cite{6, 23} to define an ontology. In the following, a selection of existing works is discussed. METHONTOLOGY \cite{14}, On-To-Knowledge Methodology (OTKM) \cite{24} and DILIGENT \cite{12} constitute the basis for many subsequent proposals. For instance, NeOn \cite{18} emphasizes in reusing existing resources for building a collaborative ontology instead of starting from scratch such as in METHONTOLOGY \cite{14}.

In relation with the reusable resources related to the automotive domain, in 2018 a survey of existing ontologies for transportation was carried out \cite{16} where several approaches are studied and compared. Among them, the ontology for road traffic management \cite{3} presents bidirectional axioms (‘doesAction’ and ‘isActionDoneBy’) that relate classes with driving actions. There are others that also consider attributes by introducing additional axioms to the ontology.

UPON \cite{1} was published in 2005 as a proposal that takes advantage of the Unified Software Development Process and the Unified Modeling Language (UML). The proposed methodology is based on the semantic languages created by the World Wide Web (W3C), RDF and OWL. These XML-based syntaxes allow the representation of knowledge as triples, i.e. a 3-entity statement in the form of subject-predicate-object expressions. This atomic structure forms a directed graph, hence ontologies can be defined as graphs, where each class is a node (vertex), and it is connected to other classes via properties or relations (edges). Some works have considered the use of graph databases to implement RDF stores \cite{19} or to build an ontology for an automated vehicle’s context model \cite{25}.

However, there is not yet a de jure standard to build or design ontologies in the automotive domain. Furthermore, the proposed methodology bases the construction and representation of the ontologies in graph databases. Among the analyzed references, and to the best of our knowledge, sound developments based on graphs have not been found, so there is not a dominant, standardised methodology based on these databases. In this work we use Neo4j as the graph database to host the ontology and the Cypher query language to interoperate with it. This database deploys the ontology as a database resource, fostering its utilisation in the new challenges of the labeling industry, such as worldwide networking (e.g. using Bolt network protocol for client-server communication), Big Data, advanced algorithms (e.g. pathfinding) and visualization applications.

3. Terminology

In this work, the following definitions have been adopted:

- **Ontology**: formal description of concepts (Class) and their relations (Axioms) according to a common understanding of experts in the domain. The definition of these elements can be completed with properties or restrictions.

- **Class**: concept of the domain represented as a node in the graph database.

- **Individual**: instance of an ontology class. In the case of automotive scenarios, a named class should be assigned to each individual of any scene. Hence, individuals of a class are defined by a unique identifier (UID) and a unique name that is specific for each analyzed case.

- **Axiom (Relationship)**: statements that are asserted to be true in the domain being described \cite{5}. They structure the ontology and provide semantic information. They are represented as relationships in the graph database.

- **Object**: a class that represents anything tangible, e.g. a person or thing. They are the main elements of the ontology and could be related with attributes or actions.
– **Action**: a class understood as a situation with a semantic meaning, happening in the scene typically related to Objects, which are either the subjects or the objects of the Action. They occur during a specific time interval (frames).

It is necessary to distinguish between intransitive and transitive actions, as they are semantically different.

1. **Transitive action**: can be naturally treated as triples, where there is a subject, a predicate and an object. For example, “a child is running in the park”, where ‘Child’ is the subject, ‘Park’ is the object and ‘Running’ will be the predicate or the Action.

2. **Intransitive action**: express status of Objects, and thus can be expressed as adjectives or verbs in present continuous form: “The car is parked”. In this example the object is not specified, but the ‘Car’ is known to be the subject and ‘Parked’ is the predicate of the sentence.

– **Event**: a class to represent anything that happens in an instant of time (frame). Therefore, any instantaneous change of state caused by an Object could be defined as an Event. These changes usually cause a new occurrence and depending on the duration this can be defined as a new Event or an Action.

– **Context**: a class for elements that describe the general situation and circumstances of the scenario. Contextualizing can involve any aspect that helps the user or application define the surroundings and general conditions of the scenario.

– **Attribute**: a quality or feature of a class element or axiom of the ontology.

– **Scenario**: a quantitative and qualitative description of the situation (e.g. traffic environment), as the sequence of Actions and Events performed by Objects.

4. The AGO domain model

AGO aims to cover the main elements required to support semantic labeling and the description of automotive scenarios for testing environments. Hence, the core concepts defined in the ontology correspond to those used to structure the information in VCD and OpenLABEL. Its high-level structure can be seen in Fig. 1. On the one hand, the main superclasses of the ontology are **Object, Context, Action and Event**. These elements form the first level of classes and all other classes are derived from them. On the other hand, in VCD and OpenLABEL, the **Relation** element is required to structure the domain knowledge and semantically enrich the ontology. The RDF language model [6] defines several axioms for describing properties and relationships among named terms: ‘rdfs: domain’, ‘rdfs: range’, ‘rdfs: subClassOf’.

Additionally, some non-standard relations are proposed in AGO for the purpose of covering the needs of the description of automotive scenarios.

– ‘isSubjectOfAction’ and ‘isObjectOfAction’: these axioms are defined for actions expressed with transitive verbs. Transitive actions can be naturally treated as triples from a language point of view. Nevertheless, transitive triples do not allow relating the action with other nodes of the ontology (because the action in a transitive triple is the predicate or relation between subject and object, and thus it is not a class). Therefore, transitive actions are unwrapped as 2 related RDF triples, where the action is a class and the relations are ‘isSubjectOfAction’ and ‘isObjectOfAction’. In the case of the intransitive actions, a unique triple is generated relating the element that performs the action and the Action itself.

– ‘isSubjectOfEvent’ and ‘isObjectOfEvent’: as for the Actions, the Events can be also distinguished
among transitive and intransitive. Therefore, the same type of relations are defined in AGO for these elements. One describes ‘who’ performs the Event and the other ‘who/what’ is affected by it.

- ‘causes’: Events are occurrences that happen in a time instant and they usually trigger another Event or Action. Therefore, this axiom is adopted to relate Events with Actions and represent this effect.

In order to provide the ontology the capability of representing spatio-temporal information by relating the different classes two additions have been performed. On the one hand, Allen’s temporal relations [22] are adopted (e.g. ‘meets’ as a relation used to define the timeline of the scenario by relating Action and Events in temporal order).

On the other hand, for defining the spatial relations among the elements (e.g. required to construct a complete description of the road network):

- ‘isPartOf’: describes a spatial relation between the different Objects (i.e. “Lane” - “isPartOf” -> “Road”)
- ‘isConnectedTo’: this relation is defined to relate all the spatial objects describing the a network of objects. Concatenating these static objects (i.e. “Road” - “isConnectedTo” -> “Intersection”) is required to formally represent the road network in OpenDRIVE\textsuperscript{15} format.

Furthermore, spatial relations among objects in the scenario can be further specified by: ‘behindOf’, ‘inFrontOf’, ‘leftOf’, ‘rightOf’ and ‘middleOf’.

5. Methodology: AGO construction

This section comprises the know-how acquired during the process of constructing AGO. Fig. 2 shows the pipeline of the methodology for the construction of a domain ontology.

Table 1
Summary of the analyzed datasets for defining the taxonomy of the ontology.

| Dataset          | Year | Object Classes | Action Classes |
|------------------|------|----------------|----------------|
| nuScenes         | 2019 | 23             | 5              |
| H3D Honda        | 2019 | 8              | -              |
| LyftLevel5       | 2019 | 9              | 18             |
| Waymo            | 2019 | 4              | -              |
| ApolloScope      | 2019 | 65             | -              |
| Berkeley-Deep-Drive | 2020 | 40            | -              |
| Audi A2-D2       | 2020 | 52             | -              |
| Mapillary-Vistas | 2020 | 66             | -              |
| SafetyPool       | 2020 | 126            | -              |

5.1. First phase: Definition of the scope and Knowledge acquisition

The first stage is an analysis phase and will directly influence the whole process. It aims at setting the concrete scope of the use case. This will determine which objects are included in the taxonomy of classes. Along with this, correctly identifying the requirements of the use case will govern the inclusion of further elements into the ontology. This step requires having a deep knowledge of the domain of interest and can be the most expensive phase in terms of person-hours. Therefore, in this case a domain expert of the automotive area (i.e. traffic manager, Intelligent Transportation Systems (ITS) researcher, etc.) should perform the task. It is not mandatory for this role to have ontology modeling knowledge at this stage.

This phase also involves acquiring the knowledge for defining the elements of the ontology. In this case, several large-scale open datasets for the development of AD technologies and the Safety Pool taxonomy have been analyzed. The number of elements is summarized in Table 1. The goal is to define generic classes that cover as many elements of the analyzed resources as possible to enhance interoperability while keeping sufficient level of detail for the applications. Thus, the output should be a domain-specific hierarchically structured knowledge graph.

In general, these datasets do not define classes in a hierarchy but often as one-level list. In addition, only two of the listed datasets cover a few classes related with maneuvers (Action classes in AGO domain). However, actions are presented as special attributes of the corresponding objects. Because of how they are defined, these datasets do not consider the need of making a distinction between Action and Events. Moreover, they do not present semantically-relevant relations among these elements and the objects. Taking

\textsuperscript{15}https://www.asam.net/standards/detail/opendrive/
into account that most of these datasets do not consider the traffic maneuvers, the ‘ALKS Regulation UN R157’ has been included in AGO so that the ontology gathers the critical scenarios listed in the regulation as Actions along with some additional Object classes and properties. As a result, AGO is an ontology which offers flexibility for present and future labeling needs, supported by the established mechanism to connect Objects, Actions and Events through Relations. AGO also includes in its current form a rich set of object classes, easily extensible, and an equivalently wide number of actions and maneuvers that can serve as a basis for multiple applications in the domain (e.g. test coverage analysis, online data recording, digital twins, etc.)

5.2. Second phase: Build the taxonomy of classes

5.2.1. Technologies and Tool

The Semantic Web focuses on enabling machines with the capabilities of providing formal structured information using the encoded knowledge from ontologies [27]. There are several languages that enable formalizing and encoding knowledge. In our case, the chosen ontology representation language is The Web Ontology Language (OWL) [23]. OWL is developed as a vocabulary extension of the Resource Description Framework (RDF) specification. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes [21].

There are several tools specifically designed to create and manage ontologies: Protégé, Protégé Web, Fluent Editor (FE), OWLGrEd, etc. For this approach, the ontology has been defined with Protégé as this tool allows constructing complete ontologies quite intuitively. Likewise, the ontology can be exported into different ontology representation languages including RDF and OWL. This tool also provides the user with a Taxonomy Tree viewer (see Fig. 3) and other visualization tools, such as OWLViz that generates a diagram of the selected elements in different layouts.

This tool also provides the means to include Annotation properties which comprises ontology metadata and general properties of the classes. For the metadata, the versionInfo (owl:versionInfo) is defined as a built-in annotation property of OWL [23]. Furthermore, some of the Dublin Core Metadata Initiative (DCMI) [8]

terms have been adopted: Contributor, creator, description, title, date submitted, license and publisher. In addition, to complete the description of the classes a brief description and a label are included as defined by the RDF schema [6]: rdfs:comment and rdfs:label.

5.2.2. Ontology construction

The domain ontology is constructed with the common classes identified among the datasets in the first phase, where the analysis is done with the objective of identifying the main terms and the related concepts. Thus, this phase involves the following tasks:

1. Identification of the common classes among the datasets
2. Define upper level classes for structuring the hierarchy logically
3. Include every class defined in the datasets (manage synonyms)

The defined ontology provides a shared understanding of common concepts (Classes) among the main automotive open datasets. Therefore, it can be considered as a domain ontology. The general classification of the classes is done according to the element types de-
defined in the AGO domain model description Section 4. Hence, the upper level consists of 4 core-concepts as depicted in Fig. 1. Besides, it is constructed top-down using the *Containment Relationship Type* \cite{26} (‘rdfs:subClassOf’) and according to the hierarchy defined on the datasets. Consequently, most of the included classes appear in one or more of the analyzed datasets. After this process, the hierarchy of classes evolves from the simpler tree or plain lists from the source datasets into a more complex graph where each class may have several parent nodes.

5.2.3. Knowledge representation language and knowledge graph structure

In the OWL Reference documentation, the W3C states that an OWL ontology is an RDF graph, which is in turn a set of RDF triples \cite{23} (subject - predicate -> object). Therefore, some notations from the data-modeling vocabulary defined by the RDF Schema and the OWL language principles have been adopted for the purpose of this work \cite{6}:

1. Classes are identified by an Internationalized Resource Identifier (IRI). Also, each Class is represented by a lexically meaningful Uniform Resource Identifier (URI) that will be unique for each entity.
2. At the beginning of the RDF/XML document an ontology header is included with the defined base URI.
3. A node representing the set of all individuals is defined with the class extension of owl:Thing.

All listed characteristics are automatically considered by Protégé when including new Class entities in the hierarchy. To complete the description of the elements a *description* and a *label* are included as annotations which results in a class representation, as depicted in Fig. 4.

The proposed structure of AGO is a directed-graph since this configuration provides the means of representing high-order semantic relations as RDF triples with basic elements of this non-SQL datasets. In terms of the triples, the *subject* and *object* are represented by ‘Class’ labelled nodes. Data properties (owl:DataProperty) are also included as nodes under the ‘dataProperty’ label. They serve as attributes to the classes according to the OWL specification. To semantically cover a traffic scenario several complex relationships are required to link different elements. The user-defined axioms presented in Section 4 are included as object properties (owl:ObjectProperty) which according to the OWL (RDF/XML) Structural Specification \cite{5} it is the construct defined to connect pairs of individuals with user-defined relationships. Therefore, the graph is populated with them as nodes with the ‘objectProperty’ label to represent the *predicate* of the triples. Both property type nodes are related using the ‘DOMAIN’ and ‘RANGE’ edges. This explanation is illustrated in the graph snippet of Fig. 6.

5.3. Third phase: database building in Neo4j

The third stage implies building the knowledge-graph database and making the ontology available for its use via programmatic interfaces. To this end, a database to store the data is selected. In this case, Neo4j\textsuperscript{18} (*neo4j-community-4.2.0*) has been chosen as the database for storing the ontology. To represent the ontology as a graph in Neo4j following the standards, the neosemantics plugin (n10s)\textsuperscript{19} is required. This enables importing and exporting the ontology graph from and into OWL files for further use. Using neosemantics *Release 4.1.0* a Cypher query can be used to import the OWL ontology as depicted in Fig. 5.

The imported data is further processed by including specific Neo4j-labels to classify the nodes according to the core concepts. Grouping nodes with tags related to the first level of classes in the hierarchy (i.e. Action, Object, etc.) helps optimizing queries to the ontology since the consultation is done to a smaller subsets of nodes. Consequently, the ontology-based applications

\textsuperscript{18}https://neo4j.com/neo4j-graph-database/
\textsuperscript{19}https://github.com/neo4j-labs/neosemantics

Fig. 5. Main cypher query used to import the ontology.
developed in later works should be optimized in terms of ontology consultation.

The resulting graphs’ main characteristics are summarized in Table 2. AGO has 523 nodes and 1365 relationships in total. Some of the class nodes have related properties to provide the user with a complete semantic information about each element. These are known as property keys in Neo4j.

When including the elements into Neo4j, the features of each element are added as node properties. Table 3 summarizes the included property keys with the related OWL syntax equivalence.

The taxonomy tree for the main classes in Prótegé and OpenLABEL differs, since the relations are not defined as a Class type in the graph representation. As for the user-defined axioms, these are defined as object property nodes and connected with the required elements with the ‘DOMAIN’ and ‘RANGE’ relationships to build the RDF triples. Hence, the subject of the triple is represented by the domain and the object by the range terms. Fig. 6 represents the ‘Lane’ - ‘is-PartOf’ -> ‘Road’ object property triple as a graph. Besides, Fig. 6 also depicts how the ‘curvature’ attribute is related to the ‘Road’ class following the data property syntax definition.

5.4. Final phase: consumption of the ontology

The last phase is the consumption phase, as the ontology can be a helpful tool for multiple applications. Hence, in the next section some possible use cases have been briefly mentioned as a means to evidence the capabilities provided by the defined ontology.

6. Use Cases

This section summarizes two different but related use cases in the automotive domain. The first one proposes the utilisation of the ontology to guide the creation of configuration files for semantic labeling applications. The second is about creating a database of graphical scenario representations from real-data labels and from expert knowledge.

6.1. Semantic labeling

Labeling is the process of creating descriptions of the content of some data. For images or other sensorial data, labels are typically spatio-temporal entities that determine the presence of objects or actions in the reality captured by the sensors [17]. With the emergence of Deep Learning (DL), labeling has become a major activity for automakers and providers of electronics. With DL, a sufficiently large and rich dataset containing sensorial data and labels can be used to train models which learn from the dataset and predict labels on previously unseen data. This ability has triggered the creation of many Advanced Driver Assistance Systems (ADAS) and Autonomous Driving (AD) functions.

As a consequence, datasets have become a critical asset for players in the automotive market. Labels are frequently defined using function-specific or customized taxonomies, and lacking a global or universal hierarchy. Datasets are usually non-compatible and difficult to merge due to semantic inconsistencies between the used terms.

In this sense, AGO, as a domain ontology, can serve as the core of a data translation function which maps relations between (otherwise non-interoperable) datasets, as depicted in Fig. 7 and Fig. 8.

This type of application allows finding the relations among different datasets and translating the terms to the needed terminology. The translation can be done automatically or manually, with user interfaces showing a graph visualization and navigation capabilities.

Furthermore, the ontology could be part of a knowledge management system and used before the labels are created, at the annotation or labeling stage (see Fig. 9). Concretely, serving as a database with a formal definition of a unified understanding of the terminology related with the use case. This way, the database can be used for generating structure or configuration files that can guide a labeling application or process to produce labels not only in the expected format (e.g. OpenLABEL), but also semantically compatible with
Table 2
Main characteristics of the Ontology

| Ontology Name | AGO |
|---------------|-----|
| Version       | 1.0.0 |
| Neo4j-labels  | Context, Object, Action, Event, dataProperty, objectProperty, metadata |
| Number of Classes (Nodes) | 392 |
| Number of Objects | 297 |
| Number of Actions | 42 |
| Number of Events | 24 |
| Number of Contexts | 33 |
| Number of dataProperties | 57 |
| Number of objectProperties | 64 |
| Number of Relations | 1365 |
| Relationship types | subClassOf, subPropertyOf, DOMAIN, RANGE |
| Number of subClassOf Relations | 398 |
| Number of subPropertyOf Relations | 119 |
| Number of DOMAIN Relations | 363 |
| Number of RANGE Relations | 485 |
| Property Keys | label, name, comment, uri, type |

Table 3
Property key summary and OWL syntax equivalence.

| Property key | Description | OWL syntax |
|--------------|-------------|------------|
| uid          | A unique identifier is included by default to each graph element | - |
| label        | Human-friendly label. Defined for all elements | rdfs: label |
| name         | The name of each class is given in CamelCase. Defined for all the elements | - |
| comment      | A brief description of the defined element nature. The user should review the comment to understand how the elements are understood in the automotive domain. | rdfs: comment |
| uri          | http://vcd.vicomtech.org/ontology/automotive# + {name} | rdf:about |
| type         | Represents the type of data that each dataProperty element should have, which is related with the properties of the classes. For example, the 'color' attribute should be a string while 'height' an integer instead. | Datatypes |

the ontology. Also related with these files, both the structure and the terms defined using natural language could be validated with a content-checker application. This way, even if there are different users working with the tool or defining several use cases, the terms will be chosen from the ontology and will remain the same for every use case. For instance, in the case of web-based annotation tools [2], its functionality could be boosted by including relations among the annotated objects. In addition, the previous annotations and configuration files could be checked according to the vocabulary defined in the ontology enhancing interoperability by guaranteeing a common understanding of the domain concepts.

6.2. Graphical Scenario Representation

Scenario representation is also one key aspect of ADAS/AD development and testing. Rich and realistic scenario representations can lead to the generation of simulated environments that can be utilised in virtual testing procedures (e.g. Hardware-in-the-Loop simulations). Scenarios might include description of the participants of a road or driving scene, including their interactions, spatio-temporal relations, etc.

Scenarios shall be generated from expert knowledge, i.e. from high-level descriptions of how the situation should be, or from real data, i.e. from semantic labels obtained from annotation processes.

In this use case we have implemented both approaches, creating synthetic scenarios from the ALKS
Fig. 7. Example of the Waymo dataset mapping to AGO. The matched classes will represent the common terms among the different datasets and thus, the translation could be done finding these equivalences.

Fig. 8. Using AGO ontology for translating of heterogeneous labels (e.g. from different datasets.)

regulation\textsuperscript{20}, and real scenarios from the KITTI dataset\textsuperscript{21}. These scenarios have been created by first using VCD toolkit to create the RDF entries in OpenLABEL format, which is flexible enough to host high-level ac-

\textsuperscript{20}https://undocs.org/ECE/TRANS/WP.29/2020/81
\textsuperscript{21}http://www.cvlibs.net/datasets/kitti/eval_tracking.php

Fig. 9. Using AGO ontology to specify taxonomies and configuration files before the labeling stage to produce semantically compliant content.

Fig. 10. Diagram of the example KITTI scenario elements and relations illustrated over time.

1. \texttt{crl_text}: a textual description of the scenario in a Controlled Natural Language (CNL).
2. \texttt{date_db}: date and time of the latest update of the node.
3. \texttt{scenario_uid}: the scenario uid is composed by the information source and a numerical id.
4. \texttt{schema_version}: the VCD version in which the imported information was represented in.

Continuing with the scenario representation illustrated in Fig. 11, the upper nodes represent the core static elements that correspond to the upper items of the VCD JSON schema. Hence, the metadata information is included as node properties of the center node
of the representation. The working example presented in this section is represented by the schematic diagram in Fig. 10. Additionally, the scenario is graphically depicted in Fig. 11. The depicted example includes some individuals and these nodes have the following information included as node properties:

1. `frame_intervals`: the start and end frames for each node.
2. `name`: the name of the individual in CamelCase given as the class name plus a number used to list the individuals with the same type.
3. `type`: the name of the corresponding ontology class.

The individuals are presented with semantically meaningful relations among them. These relations could be easily translated into the form of RDF triples. On the one hand, taking into account the pre-defined `subClassOf` containment relationship.

On the other hand, considering the AGO domain axioms, the list of the extracted triples for the example in Fig. 11 can be extended. Taking all these triples into account, a translation into a NL textual description can be easily done. Both the list of triples and the NL textual description are presented in Table 4. At this stage of the pipeline, the resulting data would correspond to a functional scenario [28], which can be extended to obtain a logical scenario by considering the data properties defined in the ontology and included with the `hasAttribute` relation to the scenario representation.

7. Resources/Results

The example ontology produced following the proposed methodology is available online in RDF format. The automotive ontology is composed by 392 class elements classified in three main groups using Neo4j labels: `Object`, `Context`, `Action` and `Event`. The definition of each element is completed with annotation properties. In addition, the file contains 1365 relationships of which 398 represent hierarchical relations among the classes (and so defining a graph hierarchy). AGO can be used as a top-level ontology and reuse it as a starting point to build new domain or application ontologies. Furthermore, SWRL (Semantic Web Rule Language) rules are included into the ontology file to extend the axioms of the scenarios by inferring knowledge. They can be also used to validate that the inclusion on the individuals is correct. Nevertheless, these rules are not imported into Neo4j and therefore, the usability of them is not further extended in this paper.

The scripts used to build the ontology and scenario databases in Neo4j and other additional material such as ALKS and KITTI functional scenario files can be found at the Github repository.

7.1. Comparison with existing Ontologies

Different available ontologies related to the automotive domain have been analyzed in order to find the gaps that need to be covered for semantic labelling and scenario representation. Starting with the Transport Disruption Ontology [7] mentioned in Section 2, an exhaustive list of hierarchically classified events that could cause disturbances in traffic scenarios is

---

22https://vcd.vicomtech.org/ontology/automotive
23https://www.w3.org/Submission/SWRL/
24https://github.com/Vicomtech/video-content-description-VCD/tree/develop/ontologies
Table 4
RDF Triples converted into NL textual description for the KITTI dataset example scenario.

| Scenario UID | vcd430_kitti_tracking_0000_actions_static |
|--------------|-------------------------------------------|
| RDF Triples  |                                           |
|              | "EgoVehicle" -subClassOf-> "Object"        |
|              | "Car2" -subClassOf-> "Object"              |
|              | "Road1" -subClassOf-> "Object"             |
|              | "Lane1" -subClassOf-> "Object"             |
|              | "Lane2" -subClassOf-> "Object"             |
|              | "Pass1" -subClassOf-> "Event"              |
|              | "DrivingStraight1" -subClassOf-> "Action"  |
|              | "Lane Changing1" -subClassOf-> "Action"    |
| RDF Triples  |                                           |
|              | "Lane1" -isPartOf-> "Road1"                |
|              | "Lane2" -isPartOf-> "Road1"                |
|              | "EgoVehicle" -isSubjectOfAction-> "DrivingStraight1" |
|              | "Car2" -isSubjectOfEvent-> "Pass1"         |
|              | "EgoVehicle" -isObjectOfEvent-> "Pass1"    |
|              | "EgoVehicle" -isSubjectOfAction-> "LaneChanging1" |
|              | "Lane1" -isObjectOfEvent-> "LaneChanging1" |
|              | "Pass1" -causes-> "LaneChanging1"           |
|              | "DrivingStraight1" -meets-> "Pass1" -meets-> "LaneChanging1" |

NL Description
The ego-vehicle is driving straight in lane1, which is part of a single-way two-lane road. When another car passes the ego-vehicle, then it starts lane changing into the other lane of the same road, lane2.

Presented. "Agent" is defined to be the subject of the events, however, the only objects defined as subclasses of it are "Person", "Group" and "Organization". The scope of the Transport Disruption Ontology differs from the interests of AGO, consequently this ontology does not cover the whole range of actions and objects related with traffic scenarios. Nevertheless, it could be used to extend the classes re-using existing developments. Further, the listed 'event' classes have related temporal objects based on the OWL-Time ontology which allows distinguishing among occurrences that happen instantaneously or during a time range. This means that they have somehow identified the need of making a distinction between Action and Events, however it is implemented implicitly and not explicitly. Hence, some object properties defined in relation with the 'time' classes work as Allen's temporal relations do.

Most of the published approaches cover completely different scopes among the automotive domain, therefore, they lack many classes and properties to cover the requirements of scenario representation and semantic labelling. One example could be The Automotive Ontology (AUTO) [15] created by the W3C Automotive Ontology Community Group, that only covers classes related with popular cars, buses and motorcycles. Continuing with TTI Core [13], this is a layered approach presented as 3 Core Ontologies for Safe Autonomous Driving: Car, Control and Map ontologies. Each of them covers a minimum part of the domain related classes and despite not covering space-time relationships, it presents elements in order to relate objects with map elements.

The Ontology for scenarios for the assessment of Autonomous Vehicles [9] is instead the method that most resembles AGO, since it makes a clear distinction between Action and Event classes. However, the relationships defined to relate classes are not clear since the ontology is not publicly accessible and the information about it is scarce.

8. Conclusions
Most existing ontology-building approaches in the automotive domain do not use graph based tools as part of their pipeline. The presented methodology is based on Neo4j, a graph-database that provides flexibility to easily modify and update the ontology with new information. This is a key feature when developing ontology-based approaches, since new scenarios will generate new individuals and in turn, these ele-
Table 5
Comparison of AGO with existing ontologies.

| Name                               | Description                                                                 | Open Access | Format         | Spatio-temporal relations | Action support | Event support |
|------------------------------------|-----------------------------------------------------------------------------|-------------|----------------|---------------------------|----------------|---------------|
| Automotive Global Ontology (AGO)   | Automotive domain ontology for traffic scenario representation and semantic labeling applications. | yes         | OWL (RDF syntax) | yes                       | yes            | yes           |
| The Transport Disruption Ontology  | A formal framework for modeling travel and transport related events that have a disruptive impact on an agent’s planned travel. | yes         | OWL (RDF syntax) | no                        | no             | no            |
| The Automotive Ontology (AUTO)     | The essential real world objects related to popular vehicles like cars, buses and motorcycles. | yes         | RDF            | no                        | no             | no            |
| Ontology for scenarios for the assessment of AVs | An ontology for describing scenarios in the context of the assessment of the performance of an AV. | no          | UML            | yes                       | yes            | yes           |
| TTI Core                           | A machine-understandable Knowledge Base for autonomous driving vehicles constructed using 3 core ontologies: map, control and car ontology. | Under license | OWL (RDF syntax) | no                        | no             | no            |

In this work, the area of interest was the automotive applications of semantic labeling and scenario-based testing. Hence, a formal description of the ontology that covers as many driving scenes as possible has been done. The pipeline is based on the taxonomical structure of classes presented in the main automotive datasets with added expressivity and semantical load by including new relations. As result a reusable top-level automotive domain ontology has been defined and named as AGO (Automotive Global Ontology).

The methodology is defined with accessible tools and steps that do not require significant technical background. The objective is to provide the means to construct and take advantage of ontologies to as many user profiles as possible.

The know-how presented in this paper about the construction of a domain ontology, and the two selected use cases, has been made available to the ASAM standardization group25 for the development of the OpenLABEL and OpenXOntology standardisation projects (to appear 2021-2022)26, to foster the contribute to the automotive industry and scientific community.

Acknowledgements

This work has received funding from the European Union’s H2020 research and innovation program (grant agreement nº 824309, project HEADSTART).

References

[1] A. De Nicola, M. Missikoff and R. Navigli. (2005). A Proposal for a Unified Process for Ontology Building: UPON. Lecture Notes in Computer Science, 655–664. doi:10.1007/11546924_64.
[2] A. Mujika, A. Dominguez, I. Tamayo, O. Senderos, J. Barandiaran, N. Aranjuelo, M. Nieto, and O. Otaequi, “Web-based Video-Assisted Point Cloud Annotation for ADAS validation,” Web3D 2019 – The 24th International ACM Conference on 3D Web Technology, Los Angeles, CA, United States, July 26-28, 2019 (DOI: 10.1145/3329714.3338128).
[3] A.J. Bermujo, J. Villadangos, J.J. Astrain, A. Córdoba (2013) Ontology Based Road Traffic Management. In: Fortino G., Badica C., Malgeri M., Unland R. (eds) Intelligent Distributed Computing VI. Studies in Computational Intelligence, vol 446. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32524-3_14

25https://www.asam.net/

26https://www.asam.net/project-detail/asam-openxontology/
