The Mid Level Data Collection Ontology (DCO)
Generic Data Collection using a Mid Level Ontology

Joel Cummings and Deborah Stacey
School of Computer Science, University of Guelph, Guelph, Ontario, Canada

Keywords: Mid Level Ontology, Data Collection Ontology (DCO), BFO, Data Collection, OBO Foundry, Foundational Ontology, Upper Level Ontology, Domain Ontology.

Abstract: Capturing data through an ontology is a common goal where instances exist as datums mapping to universal terms defined in an ontology. Currently these ontologies lack a shared conceptualization for data collection terms. We propose a mid level Data Collection Ontology (DCO) that defines data collection terms in a domain agnostic way enabling extension for domain ontologies to build off of. Such an ontology should provide reasoning support and enable automated error detection required by all data collection ontologies. By using the Basic Formal Ontology (BFO) as its base it enables existing OBO foundry ontologies to extend the proposed ontology in their design allowing existing domain level ontologies an entry point.

1 INTRODUCTION

Collecting data is a common purpose for an ontology whose terms, descriptions, and relationships describe universal categories that collected data are arranged under as instances. Due to the requirement of domain terms these ontologies are created at the domain level meaning they only seek to define terms that reflect the particular domain they operate in, ignoring hierarchies and more general terms that may apply to other areas. The result is an ontology that defines data collection with a domain specific view of the world.

We define *ontology* as a shared conceptualization that should seek to define terms in their most formal regard and should not use terms that are specific to particular areas wherever possible (Gruber, 1995). The data collection components of domain specific ontologies provide little in the way of reuse potential and violate the idea of shared conceptualization in defining data collection terms. Ontology developers should therefore strive to produce solutions that enable reuse among other ontologies instead of redefining terms and patterns that can be defined once and shared (Gruber, 1995). It is for this reason upper level ontologies exist that seek to define terms that are necessary for any ontology. An example is the Basic Formal Ontology (Bas, 2017) which defines domain neutral terms that can be used for all ontologies due to their high level of formalism and domain agnosticism. BFO starts by organizing terms by where they sit in the world based on if they exist in time space.

Therefore, upper level ontologies seek to serve all ontologies regardless of domain or purpose. Mid level ontologies are a level deeper and build off of upper level ontologies by providing terms that apply to ontologies of a large domain or similar purpose and put these terms in the appropriate hierarchical space defined by the upper level ontology they extend. This provides a stepping stone for domain level ontologies through allowing them to share more specifically related terms with other domain level ontologies and increasing the potential for reuse that upper level ontologies offer. In the context of our problem we might say that our domain or purpose is data collection and we seek to define terms to allow data collection regardless of domain. In this case the definition of mid level ontologies is what we are interested in. Thus we ascertain that our problem centers around the creation of a mid level ontology that serves to define the data collection domain. We argue that the creation of a mid level ontology for the purpose of data collection will help foster reuse and enable faster creation of domain level ontologies that collect data through instances.

In this paper we present our idea for what a mid level Data Collection Ontology (DCO) will look like, where it fits in the ontology hierarchy and how it will remain domain independent. Specifically, we will look at particular definitions and where the DCO fits into the existing ontology framework.

Cummings J. and Stacey D.
The Mid Level Data Collection Ontology (DCO) - Generic Data Collection using a Mid Level Ontology.
DOI: 10.5220/0006497501750182
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KEOD 2017)*, pages 175-182
ISBN: 978-989-758-272-1

2 BACKGROUND

Ontologies that are designed for reuse in a more general form will help to frame our problem and provide terms as a starting place for our design. In this section we discuss different types of ontologies and the level of concern they have for reuse to determine if and where existing designs or ontologies can be reused by a generic data collection ontology. We then focus on ontology categorization in the context of where our problem is best tackled keeping in mind our desired high level view of data collection. Finally, we summarize by drawing conclusions on existing designs and what needs to be done in terms of defining generic and reusable ontologies.

Ontological research has become widespread in the design of information systems. Recently the desire for ontologies to span and integrate different views of a domain and even across domains has come to fruition (Mascardi et al., 2010). The development of these ontologies provides the opportunity for systems to integrate and become interoperable allowing for information sharing (Herre, 2010) (Mascardi et al., 2010). In this case an ontology acts as a bridge between systems unifying information (Herre, 2010) and allowing systems to communicate through the ontology using their standard language and message passing techniques. Unifying data allows the key components of one or more domains to be captured and shared among ontologies that further define a particular domain. The ability for an ontology to capture a particular domain is related to its viewpoint of the world, each ontology imposes a particular view which defines its ability to share information. This viewpoint is therefore what we are concerned with.

2.1 Classifying Ontologies

Domain level ontologies are ontologies that seek to capture a shared conceptualization of a particular domain. These ontologies contain domain specific terms and may only be linked to a specific application (Roussey et al., 2011). Domain ontologies are important in that they describe the type of data we seek to capture but for our problem we may not assume any particular domain to capture data from. A domain level ontology, however, could represent the end product for a system using our ontology.

A **core** ontology is linked to a particular domain but has the advantage of providing several viewpoints relating to different user groups (Roussey et al., 2011). Core ontologies are often the result of several domain level ontologies mapped together (Roussey et al., 2011). Core level ontologies represent a higher level of term generality as they seek to span and provide definitions for a wider domain or domains. From our perspective core level ontologies are certainly closer but still maintain the requirement of domain specific content within them and cannot generically be applied to any domain.

Foundational or **upper level** ontologies can be summed up with the following definition: a foundational ontology seeks to provide definitions and terms that are general to all domains. (Mascardi et al., 2007). They serve as a building block for future ontologies by enabling reuse since they define common terms that will be contained by domain level ontologies. The goal of an upper level ontology is to avoid the redefinition of common terms to allow for easier and consistent reuse of defined terms. In other words they provide a single agreed upon definition of terms (Mascardi et al., 2010) (Roussey et al., 2011). More importantly however is the fact that they are designed to support all domains which differs from core or domain ontologies that only define terms for their particular domain, likely choosing specific (overloaded) definitions over general definitions (Roussey et al., 2011).

A **mid level** ontology seeks to provide a bridge between an upper ontology and a domain level ontology by providing terms that will be common to several domain level ontologies or areas of domain level ontology (Ceusters and Smith, 2015). Therefore mid level ontologies serve a similar purpose to the upper level ontology by preventing term redefinition and providing consistent relationships but at a more specific level. This has several advantages in addition to avoiding redefinition, firstly, it provides a common understanding between derived ontologies through similar terms, structures and relations, secondly, it provides a more streamlined starting place for those new to the construction of ontologies by providing terms more closely related to their domain than that of upper level ontologies. In terms of an ontology category hierarchy the mid level ontology falls in the middle with domain level ontologies extending mid level ontologies and mid level ontologies extending upper level ontologies. The full hierarchy can be seen in fig. 1.

One might then wonder why the work put into the development of upper level ontologies has not resulted in a common ontology that is shared among all domain level ontologies. One particular reason for this is down to implementation, where languages implemented by computer scientists are based on set theory that captures abstract content well but does not capture the concrete objects and their relationships well enough to be completely generic (Degen et al.,

Upper Level Ontologies	General terms for all ontologies to be based off. Enforces high level structure.
Mid Level Ontologies	Bridge the gap between generic upper level terms to domain level terms.
Core Ontologies	Core ontologies define multiple view points or multiple domains.
Domain Level Ontologies	Ontologies that define the view a particular domain has of the world.
Application Ontologies	Local or Application ontologies define a view specific to a particular application

Figure 1: Ontology Classification Hierarchy.

2001). More recently the author of the General Formal Ontology (GFO) stated that we may not be able to meet such a lofty goal at all (Herre, 2010). However, for our purposes this is still acceptable since we must work with what is available. With that in mind we will focus on available upper level ontologies to seek a design that best meets our needs. All of our ontologies are sourced from Mascardi et al (Mascardi et al., 2007) due to their capturing of relatively recent and active implementations. For evaluation we will define a criteria that will help us to draw conclusions based on upper level ontology design, purpose, and applications.

The first criteria we define is based on the number of terms and relations in the ontology, where we prefer to have fewer of each for two main reasons. Firstly, upper level ontologies are meant to be derived into a domain level ontology and thus will have more terms and relations added over time and, large ontologies introduce performance penalties potentially resulting in an ontology that is intractable for a reasoner (Horrocks, 2005). Secondly, in terms of understandability the fewer terms a person must know to use an ontology the easier it is to get started. Furthermore, large ontologies may deter usage of the ontology altogether.

The second criteria we care about is usage and popularity. Popularity of an upper level ontology is important when considering its purpose of unifying ontologies. Also, an upper level ontology must be free from any domain specific terms or relations. Finally, we are not interested in ontologies that take the role of defining thousands of terms to satisfy a large number of domains since it is unlikely such an ontology could satisfy each domain realistically.

Ontologies considered include: the Basic Formal Ontology (BFO), the General Formal Ontology (GFO), a Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE), and the Suggested Merged Upper Ontology (SUMO).

For the sake of brevity we will focus on our choice based on the criteria defined above, although all versions were evaluated and its likely a case could be made for any of the above upper level ontologies. Our choice was the Basic Formal Ontology (BFO) which is an an upper level ontology with development starting in 1998 (Mascardi et al., 2007). It currently consists of 35 classes in version 2 making it relatively small (Bas, 2016). BFO is commonly applied in the biology domain but does exist in a number of other domains and is used in over 150 ontologies as of this writing (Bas, 2016). BFO itself contains no domain specific content, and focuses on describing objects through time and space which is common to all physical objects. It considers both abstract and concrete terms and seeks to define terms based on their lifespan as either occurrent or continuant, with occurrent defining objects that exist during a period of time while continuant objects exist throughout time (Bas, 2016).

In terms of our criteria, BFO does well through its definition of only high level concepts involving time and space, maintaining a small size making the ontology suitable for additional development and for reasoning. In terms of popularity and usage we examined resources on the web to see how many ontologies cite themselves as using each considered upper level ontology. On the BFO web page they cite well over 150 ontologies or projects using BFO (Bas, 2016). The important point here is that the projects stem from more than just the biological field which was not the case for other entries.

Based on our defined criteria, BFO fairs best which is why we will focus on discussing BFO and why it best fits our needs. The Basic Formal Ontology is in its second major version therefore we will focus on that version in discussion of terms and structure although the first version is quite similar (Bas, 2016).

2.1.1 Mid Level Ontologies

BFO provides a starting point for the creation of an ontology but does not give direction about where to stop development which can go to various levels (see section 2.1). We propose a mid level ontology as the design target for the problem and in this section discuss why that choice best reflects the problem, our definition of ontology, and works with the chosen upper level ontology. We will start at examining how the problem fits with this design as well as discussing downsides to the design.

Mid level ontologies seek to define a domain that is at a very high level and span multiple ontologies. They therefore are generally independent and define terms at a high level to avoid conflict with ontologies

that will extend them. This fits well with our problem since it is expected that our solution will form the basis of a domain ontology but not be exhaustive in term definition. Second in terms of our definition they seek to define terms as generically as possible but also while avoiding redesigning existing ontologies and redefining terms.

Another important part of mid level ontology compatibility is the source ontology. The OBO foundry provides the framework and existing ontologies that are developed using BFO and demonstrates existing mid level ontologies that are active. This demonstrates merit to the proposed pattern as it provides concrete examples functioning with the Basic Formal Ontology (WG, 2017). Furthermore, the Basic Formal Ontology does not have a derived ontology that exists for this particular problem demonstrating a gap in existing mid level ontologies into which our solution could fit. The design of BFO has taken into consideration mid level ontologies with working examples of mid level ontologies and domain level ontologies utilizing those mid level ontologies (WG, 2017).

3 ONTOLOGY DESIGN

This section is dedicated to an overview of the Data Collection Ontology (DCO), its components, relations, and design choices that make it suitable for data collection. The DCO is designed as a mid level ontology that extends the Basic Formal Ontology (BFO) to organize and provide placement for data collection terms. The DCO seeks to provide domain independent definitions as a starting place for domain data collection developers. Due to the fact that the DCO is a mid level ontology and is in its early design it is by no means finished and is expected to change over time. Like an upper level ontology it may be found to be incorrect or lacking and will need to be updated.

With its purpose in mind we start by noting the design intentions in other words: how it is expected to be used by domain ontology developers. We then move on to discussing the components and relations of the ontology to understand why particular components exist and how they contribute to the intended use of the ontology.

3.1 Design Intentions

The design intentions are an important place to start since they set the basis for how one is expected to use the DCO. The design has a philosophy about how data collection should be performed and does so at a high level allowing for more specific work flows to be integrated. This view is based on first describing what you are collecting; these are *subjects* which represent a timeless view of your object. The DCO uses BFOs independent continuants to define subjects that describe objects as they are in concept but not as an instance that exists in time and space. Instead, your captured data are represented as instances and have a type of the subject. DCO also includes processes to capture how data is collected and what stages it goes through. This is common in data collection activities such as surveys or cyclic forms of collection. Additionally DCO places stress on types and units through the definitions of datums that capture both measures and units of measure ensuring all values are labelled appropriately. The final portion of the ontology consists of *classifiers* that are entities in BFO since they are time and space irrelevant and may be used to classify any type. In this case it was felt that classifiers should not be restricted to time or space due to their function of classifying any type.

Classifiers are hierarchies of terms over which one defines equivalence relations to define what constitutes this particular category. Classifiers are designed around the suspicions or anecdotal estimates of what range one expects data to fall into. Classifiers are designed to be populated with instances, which exist as individuals of any type in the ontology. These instances are then grouped based on the reasoner and can be queried to determine if they are of the expected type when entered in the ontology. In other words, it allows validation of the estimates or anecdotal data one has. Classifiers provide additional advantages concerning data validity in that they are non-destructive whereas traditional approaches may place strict boundaries on collected data, removing instances that do not fit. Classifiers allow *invalid* or *inconsistent* data to be filtered but not permanently removed if an inconsistency in classification is detected. This supposes there is a *dynamic* aspect of the ontology that over time will be shaped by the instances that it collects and that definitions will be challenged.

3.2 Ontology Components

With the high level view of DCO established we can further break down its main components: Subjects, Processes, Data Qualities, Classifiers, and Meta Data. In this section each of these terms are defined. We then move on to using the components with the defined object and data properties to form a working example of the DCO. Due to the main premise of the DCO, or any mid level ontology, the DCO only defines terms at a relatively high level.

Table 1: Object Relations.

Relation	Description
has part	Allows individuals to be composed of other instances. This is important where data is captured on different parts of a larger item or data is aggregated into a larger sum. Composition should not be thought of only in terms of physical objects having parts.
has measure	Measurements are considered any numerical value one captures and links to an individual. Note that this is a object property so it forces one to link to some descriptor for the value.
Subclass of *has measure*	
has measurement datum	This will be one of the most common properties as it links measurement datums to individuals so data is annotated with units.
has measurement unit	This provides a link for unit definitions to measurement datums.
has time stamp	Links a time value to some measurement datum that contains some time unit allowing a universal way to save time in an ontology.

Table 2: Object Relations Continued.

Relation	Description
contains process	Links a process to an object. For example, some subject may go through some process that data is captured on. The data collection may itself be a process and have a relation to another process.
has quality	Allows instances to possess particular qualities or require particular qualities on data being classified.
has object control	Used for objects that act as a control. For example, in a process something may be a terminator.
branches to	Supposes that an instance in a process will branch to another instance when it has completed. Allows for order to captured.

3.2.1 Classes

Subjects represent what data is being collected from or about. Subjects can be either physical objects or concepts, meaning types can be either material or immaterial. Subjects are designated as independent continuants, meaning they should only represent high level subjects. For example, if we are surveying people then the subject may be a person and we would define person at a universal level while instances may have a relationship with the person subject, i.e. *part_to* Person but are themselves occurrent and do exist in space time. Additionally, through BFOs Role class, one can assign the roles that Subjects may operate in .

Processes fall under the BFO definition with extensions provided by DCO for convenience and allow one to support both state driven and independent processes. State driven processes require one process block to finish before another can start while independent processes can have any number of process blocks running concurrently.

Classifiers are where equivalence relations are defined to classify instances in your ontology. Classifiers are where one would normally define the range that data is expected to fall into so as to form a particular category or type. One may think of a classifier as having the ontology assign a type to an individual based on its understanding. Classifiers can be thought of as the dynamic component of the ontology. They are designed to change as data may prove them to be invalid or individuals may change if they are proven invalid based on the ontology's view of the world. For an example of how classifiers look see Table 2.

Meta Data are descriptors that exist to define a data point or complex structure that one expects an individual to contain. Meta data describes the types and units that data will exhibit allowing one to capture data in multiple formats and multiple units but have it link to the same individual type without causing confusion later. An example case of this would be if a study were conducted across North America where in Canada the metric system dominates while Americans use the imperial system. Data could be captured for the same study using different meta data classes to describe the units.

Data Qualities exist to define restrictions and set theory properties on instances that can be used as a part of the classifiers to group instances or as a part of a larger system. Examples of data qualities include boundedness, cardinality, and equality.

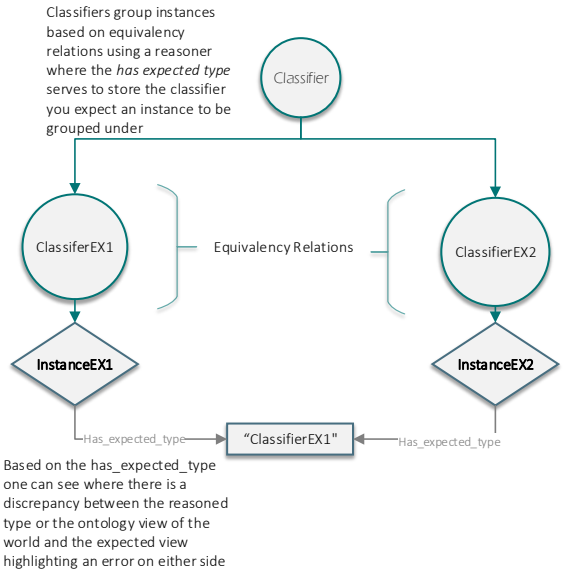

Figure 2: Classifiers.

3.2.2 Relations

In addition to the classes defined, DCO defines several high level relations that are designed to be subclassed and added to. Object relations are summed up in Tables 1 and 2 with data relations being summed up in Table 3.

4 WORKING EXAMPLE

As an example of how the design works we will construct a very basic ontology around collecting vehicle performance data with a goal of comparing the consistency of output figures against other instances (vehicles) of the same type. The first subject of our collection will be Vehicle which is the most generic object. The Vehicle subject will describe what a vehicle is composed of from the performance perspective as this is the view our ontology has of the world. For example, every vehicle has an Engine and a Transmission so we will define those as other subjects since we are interested in these components as they alter a vehicles performance substantially. Our last subjects will be the Make and Model since we need to compare like vehicles and therefore need to know who manufactured them.

Now we define the relations between our subjects. Vehicles are made up of an Engine and a Transmission so we can use composition to define a Vehicle having those parts. DCO defines the *part of* relation which allows us to produce a composite relationship. Additionally, models are produced under some Make and

Table 3: Data Properties.

Relation	**Description**
has expected property	Denotes what property this value is intended to represent. This is designed primarily for external use where a value may link with a variable.
has expected type	Denotes what type we expect an instance to be. This is intended to be used in conjunction with classifiers allowing ontology verification based on expected types. It is additionally intended to be used to link to external systems where we want an instance to link to a particular type.
has control	Represents data values that act as controls such as booleans that alter the flow of a process.
Subclass of *has control*	
can repeat	Denotes whether a particular entity can repeat such as a process block. Some processes may be cyclical.
has sequence	Denotes a sequence value that may be used to order process blocks or other entities.
has value	The base compositor for values allows an instance to be composed of particular values.
Subclass of *has value*	
has coordinate value	Used for denoting the location of instances.
has maximum	Represents a maximum expected value for an instance to have; good for creating ranges.
has minimum	Represents a minimum expected value for an instance to have; good for creating ranges.
has time value	Links time values to instances. Note that format is independent and can be any type based on the ontology design.
has percentage	Values can be captured as percentages.
has measurement value	Used to link measurement values to measurement datums.

we can consider them part of what a company produces. For our example, we can use the *has_part* for

Vehicle to the Engine and Transmission and we can subclass *part_to* to include *example_of* for Vehicle to Model while adding produces to *has_part* to state that a manufacturer produces Vehicles and Models.

Moving on to the data we would like to capture, we will define measurement datums that will capture key performance points for a vehicle. In this simple example we would like to capture the power and torque the vehicle produces so we will define some common units. Power is measured commonly using horsepower and kilowatts while torque is measured commonly using foot pounds and newton meters. These are defined as instances under Power Unit and Torque Unit measurement units respectively. Finally we create datums for power that requires a numerical value and some power unit as well as torque that requires a numerical value and some torque unit. With these measures defined we will say that a *subclass of* an Engine requires at least one of each measure using *has measurement datum* relation. Datums are also defined for fuel economy in a similar way with fuel economy units and a Vehicle having measures for city, highway, and combined fuel economy.

The design can be illustrated as seen in fig. 3 where datums and units are defined as well as subjects linking to their respective datums. This is the general structure expected for data that is to be collected on subjects.

Now since we are capturing data on vehicle performance we may define classifiers that are based on estimations of what we expect. For this example let us say we are verifying vehicles are within their rated power measurements so we will define classifiers based around manufacturer provided power and torque ratings for a particular vehicle and apply some expected variance to create boundaries. These classifiers will use range values around the power and torque measurement datums we just defined. This allows us to create classifiers around a particular model using values for the Make and Model as well as ranges for power and torque for a particular engine to group our vehicles. The greatest importance here is when populating the ontology we must use the *has expected type* relation and link to the corresponding make and model classifier to allow the data and the ontology to be validated. This is done by using the reasoner to add the type of our added instances and then querying the reasoner for the intersection of instances that are not the same type as the *has expected type* URI value stated. In other words, this presents there is an error with the vehicle that data was captured on or that the ontology has an inaccurate view of what values are valid for that vehicle. In creating classifiers it is useful to assign a name that relates to what you are capturing so *has expected type* values can be application or human generated very easily. For example, in our case we might name a classifier DodgeRam57Auto to denote that we expect this classifier to group all Dodge Ram pickups with the 5.7 litre engine and automatic transmission making it easy to to generate the URI with the data we use to populate an instance.

5 CONCLUSIONS

We have presented our Data Collection mid level ontology (DCO) which is a mid level ontology providing domain agnostic data collection terms and providing reasoning capability for derived ontologies. Based upon BFO, this ontology provides an entry point for OBO foundry ontologies that already make use of BFO as their source. Additionally through the use of the BFO and defining high level terms the DCO will achieve its goal of domain agnosticism.

The use of a mid level ontology for data collection provides several key advantages over domain level designs starting with defining common terms that all data collections ontologies will require in some form. Secondly, it sets up ontologies for reasoning and uses the reasoner for much of the typing and labelling of data to allow for automated error checking and error prevention.

In a deeper context the DCO provides greater benefit to systems at large though its classifiers which provide two key benefits. Firstly, they allow data inconsistencies to be caught through comparing the assigned type to the *has expected type* where the classifier is known to be good. Secondly, they allow estimated ranges through the ontologies world view to be validated. They may be proven to be invalid when the assigned type and *has expected type* do not match but a datum is known to be good. This allows the ontology to help perform *"error detection"* in an external system through the use of classifiers. Secondly, it can allow external systems to resolve inconsistency on the ontology itself, creating a dynamic ontology design.

Finally, the existence of a mid-level data collection ontology based on BFO also serves the existing ontological community using the BFO ontology who have a domain ontology and are interested in adding data collection to their domain ontology or are looking to classify existing ontology instances.

For the full Data Collection Ontology, please email either of the authors who will provide it to you in various formats.

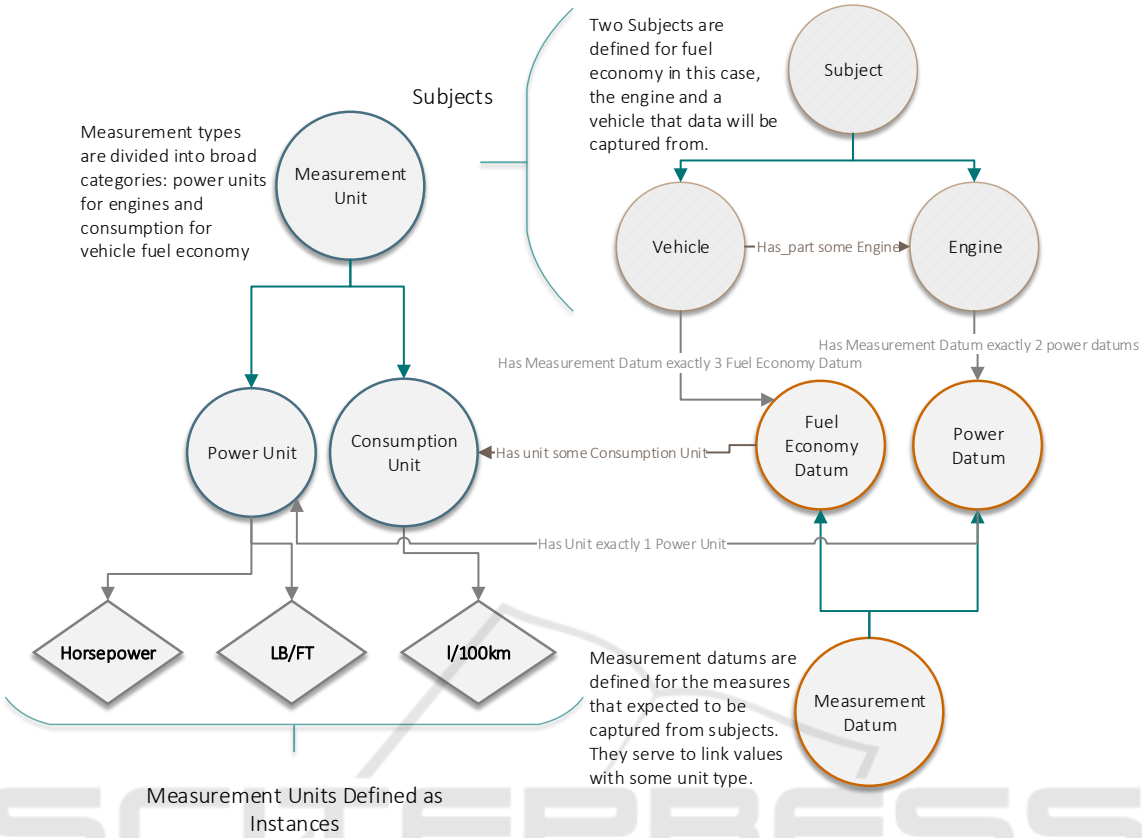

Figure 3: The Vehicle Ontology.

REFERENCES

(2016). Basic formal ontology (BFO) — users. http:// ifomis.uni-saarland.de/bfo/users. (Accessed on 02/12/2017).

(2017). Basic formal ontology. http://www.obofoundry.org/ontology/bfo.html. (Accessed on 03/26/2017).

Ceusters, W. and Smith, B. (2015). Aboutness: Towards foundations for the information artifact ontology. pages 1–5.

Degen, W., Heller, B., Herre, H., and Smith, B. (2001). Gol: A general ontological language. In *Formal Ontology and Information Systems*. Citeseer.

Gruber, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928.

Herre, H. (2010). *General Formal Ontology (GFO): A Foundational Ontology for Conceptual Modelling*, pages 297–345. Springer Netherlands, Dordrecht.

Horrocks, I. (2005). Description logics in ontology applications. In *International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 2–13. Springer.

Mascardi, V., Cordì, V., and Rosso, P. (2007). A comparison of upper ontologies. In *WOA*, volume 2007, pages 55–64.

Mascardi, V., Locoro, A., and Rosso, P. (2010). Automatic ontology matching via upper ontologies: A systematic evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 22(5):609.

Roussey, C., Pinet, F., Kang, M. A., and Corcho, O. (2011). An introduction to ontologies and ontology engineering. In *Ontologies in Urban Development Projects*, pages 9–38. Springer.

WG, O. T. (2017). The OBO foundry. http://www. obofoundry.org/. (Accessed on 03/26/2017).