A Two-Level Abstraction ODD Definition Language: Part I*

Patrick Irvine, Xizhe Zhang, Siddartha Khastgir, Edward Schwalb, and Paul Jennings

***Abstract* — The development of Automated Driving Systems (ADSs) is driven by the many benefits they offer. However, the complexities associated with ADSs and their interactions with the environment pose challenges for their safety assurance. A key aspect during its development process is knowing the capabilities, limitations, and being able to convey them in a clear manner for various types of stakeholders. The Operational Design Domain (ODD) concept was introduced to define the operating boundaries where a system can operate safely. It is therefore a key element for the safety assurance of ADSs. Efforts have been made to define the scope and the content an ODD for ADSs should cover, however there remains the need for a common, exchangeable, executable, and human-readable format for the description. This paper presents a language for the description of the ODD of ADSs, in a textual format that leans on natural language influence. Such format is intended to be both human and machine-readable and would be relevant to end users such as regulators and systems designers. The two-level abstraction approach – a structured natural language representation and a formal representation (covered across two papers) -- has been developed to have the ability to describe complex ODD conditionalities and utilize a well-defined domain ontology to achieve rich semantics. It is aimed to support ODD related activities throughout the development cycle of ADSs (specification as well as verification and validation), while covering a diverse range of stakeholders.**

Keywords— operational design domain, ODD description language, automated driving systems, verification and validation

I. INTRODUCTION

The advancement in automated driving technologies has resulted in the increasing deployment of Automated Driving Systems (ADSs) and Advanced Driver Assistance Systems (ADASs). The move towards such autonomy offers benefits such as lowered emissions [1], increased on-road safety [2], reduced traffic congestion [3], decrease driver's workload [1], and improve traffic throughput [2]. However, due to the complexity in the ADSs and ADASs, the industry and academia have been investigating different approaches for their safety assurance [3]. It is suggested that for ADASs and ADSs, it is important to establish *"how a system fails"*, i.e., the limitations of the systems [4]. However, it is not only important to establish the limitations, they also need to be conveyed to the users in order to build public trust in such systems [5].

*Research supported by the UK Research and Innovation, Innovate UK and the Centre for Connected & Autonomous Vehicles (CCAV).

Patrick Irvine, Xizhe Zhang, Siddartha Khastgir and Paul Jennings are with WMG, University of Warwick, Coventry, CV4 7AL, UK. (e-mail: {Patrick.Irvine, Jason.Zhang, S.Khastgir.1, Paul.Jennings} @warwick.ac.uk).

Edward Schwalb. (e-mail: schwalb.edward@gmail.com).

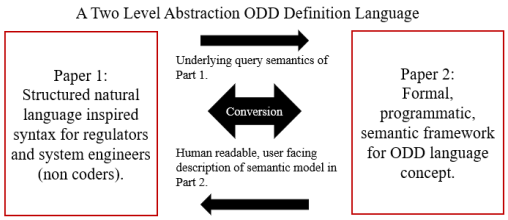

Figure 1: Two level abstraction model.

It is crucial for ADSs and ADASs to determine whether this safe operating boundary defined in the ODD is being violated or not, as it would mean triggering a transition demand or a minimal risk manoeuvre [7]. Therefore, ODD plays an important role in the safety assurance of ADSs and ADASs, and needs to be considered throughout the development cycle, from conceptual phase to verification and validation phase.

This raises the requirement that an ODD specification needs to be at the right abstraction level (and complete), and it needs to accommodate an expanding scope which might occur during the development process. In addition, various stakeholders need to access and interact with ODD specifications, such as ADS developers, test engineers and regulators. ADS developers, for instance, will benefit from a common structure for ODD definition, providing possibilities for integration into testing toolchains, along with comparability and transparency across systems and organisations. A test engineer, will appreciate this ability to easily develop toolchains through use of a common structure but will also benefit from the capability to add enough specificity to the definition that is useful in a test platform. Regulators and local authorities will appreciate the format being human-readable and interpretable for non-technical personnel, while maintaining a common format/structure that ensures consistency and comparability of definitions. Therefore, an ODD specification not only needs to have a common format for exchange and sharing, but it also needs to be quantifiable for testing purposes, and be both human/machine readable. Though some of these requirements may appear to be competing with each other, achieving a format that applies to all is essential in satisfying the needs of the industry and should be established.

Several standards have been developed in order to form a common understanding and a common scope for the ODD of ADSs, such as SAE J3016 [7], BSI PAS 1883 [6], ISO 34503 [8], SAE AVSC lexicon [9]. SAE J3016 establishes a clear definition of the term ODD and BSI PAS 1883 proposes a set of ODD attributes in the form of a taxonomy. ISO 34503 is being developed further from the ODD taxonomy in BSI PAS

1883, and SAE AVSC lexicon provides a conceptual framework and lexicon for describing the ODD. While the definitions, the attributes and the framework for ODD specification are being developed, the industry still lacks a common and open description format to enable exchange, sharing, execution and regulatory purposes.

This paper aims to build on top of the existing ODD concepts, attributes, and guidance to form a description language that satisfies the above-mentioned requirements. It reports the Part I of a two-level ODD description language concept which covers a structured natural language format with a focus on human readability. The human readable format is aimed at end users such as regulators and system designers (non-coders). The Part II [19] of the concept documents a programmatic approach of ODD definition, utilizing a query semantic format. By introducing: 1) a structured natural language format (readable to non-technical end users); 2) a formal representation (programmatic approach that is implementation ready); 3) a conversion between the two levels, a coherent ODD can be achieved which can be applied throughout the development cycle of an ADS.

II. Related Work

The definition of ODD given by SAE J3016 is as follows:

"*[The] Operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including, but not limited to, environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics*" [7].

The definition given by SAE aligns with that of BSI PAS 1883; which adds that ODD forms the foundation for the development of relevant tests that manufacturers and developers can apply consistently [6]. Through creating the ODD, we define what operating conditions that the ADS needs to handle [10].

ODD is crucial in the safe deployment of ADS, ensuring that technology has its capabilities and limitations clearly defined and readily communicated to the end user is essential for achieving *"informed safety"* [5][20]. Clear communication of an ADSs' ODD is critical for also confining the scope of a safety case as well as for verification, by restricting the case where the ADS is valid [10]. If the ADS is designed and verified against its ODD definition, it is paramount that the ADS does not leave that ODD. Therefore the ODD is positioned critically in the ADS design process from a safety perspective by establishing objective and concrete operating boundaries [10]. Blumenthal et al. [11] discussed approaches for assessing acceptable safety in ADSs, acknowledged the need for agreement on approaches to ADS safety is essential in building and sustaining consumer trust in ADSs, with ODD being positioned at the centre of this communication. Their study also comments on the concomitant effort that will be required on behalf of ADS developers and government to define conceptually and quantitatively the threshold for commercial safety through the use of ODDs [11]. The importance of ODD in ADS safety is well documented also in these literatures [12][13].

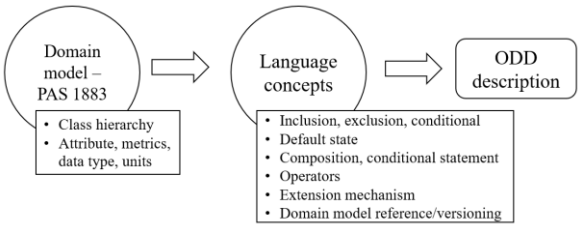

Figure 2: ODD description process diagram

As previously mentioned several standards have been developed which tackle the scope that an ODD. SAE AVSC presents a guideline for a bottom-up approach to defining an ODD [9], potential limitations of a location paired ODD approach are explored in [10]. PAS1883 presents a taxonomy for the purpose of ODD definition using their required minimum set of ODD attributes [6].

Despite its importance, there still lacks a common format for ODD description. Two examples definition formats are presented in BSI PAS 1883, one being a textual and one being a tabular definition [6]. A textual representation given follows a structured natural language format which makes for good readability, however such description needs to be structure with a formal syntax and semantics in order to capture an ODD. A tabular format on the other hand, proposes ODD as a checklist. Tabular/graphical representations can struggle to provide the flexibility needed [14]. A report by Thorn et al. [15] is consistent with the checklist approach, however Gyllenhammar et al. [10] expressed conservative views on the checklist approach due to the requirements of the details needed to represent the operating conditions in such format.

On the other hand, domain specific languages (DSLs) offer a structured format while still able to retains the structured natural language elements necessary for human readability. DSLs are now favoured in many areas, one example is the scenario description language for ADSs [16], which utilises a structured natural language format within a structured DSL framework, this satisfies the requirements from a diverse range of stakeholders. Blumenthal et al. identify that ADS developers should collaborate with state and local leaders to bring their vehicles into communities around the country [11]. This highlights the need for a human readable format in ODD description, in areas where collaboration between developers and regulators is necessary, an understandable ODD description will contribute to ensuring safety. Colwell et al. [17] note that the ODD represents not only a picture of the restrictions of an ADS but also defines the functional requirement for systems which monitor ODD; this outlines the necessity for a definition to remain executable to some degree. The necessity for a firstly implementable, and secondarily verifiable ODD definition is also referenced by Gyllenhammer et al. [10].

A gap can be identified in current literature and research, for an ODD definition language format that satisfies the regulatory and consumer requirement for human readability to establish a clear consistent communication of capability in the interest of safety. Furthermore, a language that could facilitates the above, while remaining machine readable through tool chain development, will satisfy the need of ADS

developers and validation engineers. This paper aims to propose a solution for this gap, developing a language that has heavy natural language influence (though structured) as to remain easily understandable, particularly catering for the regulatory and consumer market.

III. Development of the ODD Language

One of the main use cases of an ODD specification is to check that, during testing and deployment phases, any situations can be mapped to the ODD boundary and determined whether it is inside or outside the ODD. This requires that the ODD boundary to be binary and provides clear separation, the rest of the language concept is developed based on this key common ground. The ODD specification format enables users to define such boundary and group together a set of ODD attributes with their relations that fall within the operating boundary. This section will be split into two parts: the **domain model**, and the **language concepts**. As shown in Figure 2, the domain model provides the attributes and the basic hierarchical relations. The language concept then tackles the format of the specification, covering semantic and syntactic aspects of the language. When combining the domain model and the language concepts, individual ODD descriptions can be generated, which can express complex relations of the attributes, extend or customise the domain model, and be used for testing and regulatory purposes.

A. Domain model

Previous study has concluded that an ODD taxonomy can never be exhaustive [6] due to its complex nature. In order to provide a common set of attributes while satisfying various stakeholders, the appropriate abstraction level needs to be determined for such domain model. The idea is to have a domain model that is abstract enough such that it can be used by various stakeholders as a foundation. To further tailor towards individual's need, an extension mechanism will be introduced in the language concept to further develop and customise the domain model with traceability.

The BSI PAS 1883 [6] and ISO 34503 [8] have been developed to provide such set of recommended base attributes. While the ISO 34503 is still under development, the BSI PAS 1883 is chosen to provide the domain classes used in this paper. Please note the language concept can be used independently from any specific domain model; however, to facilitate the exchange and the sharing of the ODD specifications, it is recommended that a common domain model is used. Based on the information provided in BSI PAS 1883, five different characteristics of the taxonomy are further developed and identified, they are: class **hierarchy**, leaf node **attributes**, the **metrics** for each attribute, the **datatype** of the metrics, and the corresponding

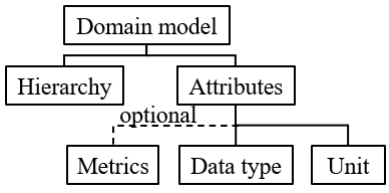

Figure 3: Main elements for defining the domain model

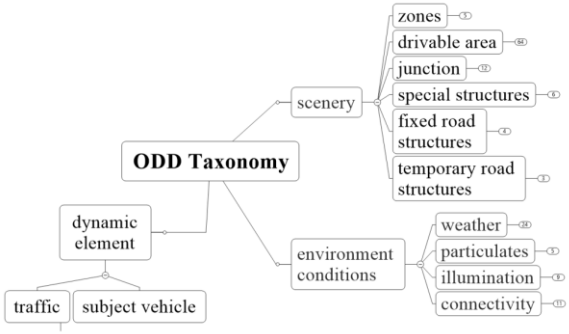

Figure 4: ODD taxonomy primary and secondary attributes

units. As shown in Figure 3, such characteristics will also be used to provide part of semantics for the language concept.

1) Class hierarchy

Figure 3 illustrates the high-level class **hierarchy** from BSI PAS 1883, this forms the scope of the ODD domain model. Three main elements are presented, which are environmental conditions, dynamic elements, and scenery. Scenery contains the necessary elements to describe the drivable area, junctions, the special structures (such as tunnel, bridge), the fixed road structures (such as streetlamp, pole), the temporary road structures (such as construction site detour) and the zones (such as interference zone, school zone). The environmental conditions contain weather (such as wind, rainfall), particulates (such as mist/fog, sand and dust), illumination condition (such as daylight, low ambient lighting) and connectivity (such as communication, positioning). In addition to scenery and environmental conditions, the ODD taxonomy also provides dynamic elements that describe the macroscopic traffic characteristics and the designated speed for the subject vehicle. Please note further details on vehicle behaviours are not part of the ODD therefore they are not considered within the scope.

2) Attributes, metrics, data type and units

Once the hierarchy of the attributes has been determined, the next step is to iterate through each of the hierarchical branches and identify the leaf-node attributes and their metrics, together with their data types and the units. Leaf-node **attributes** are the attributes that can be expressed with pre-defined metric, data type and unit from the domain model, hence they are required to be specified. An example of a leaf node attribute is *rainfall*, which sits under the parent attribute of *weather*. The ODD language allows an operating boundary to be defined within the taxonomy whereby each attribute is either included or excluded.

However, the inclusion and exclusion of attributes often needs to be linked to measurable values that relate to the real world, metrics is introduced for this purpose. The **metrics** will specify further the measuring properties that each of the identified attributes shall be described by default. One example metric for *rainfall* is *rainfall rate*. It is important to highlight that by using an extension mechanism, the user will have the capability to define customised metrics. For example, if user A wishes to measure the rainfall in terms of rain droplet size, such metric can be introduced. Table I

illustrates some example metrics as referenced to the BSI PAS 1883 with the corresponding ODD attributes.

TABLE I. EXAMPLE METRICS AND THE POTENTIAL ODD ATTRIBUTES

Attribute Metrics	*Potential ODD Attribute*
Rate	Rainfall…
Speed	Wind, Speed limit …
Visibility	Fog, Snowfall, Smoke …
Angle	Curved road, Upslope …
Radius	Curved road

Building on top of the metric, the **data type** and **units** also need to be defined, an example data type for *rainfall rate* could be *double* or *double range* (*15.00* or *15.00-17.00*), and an example of unit could be *mm/hr*. Some of the typical data types are *double, integer, string, Boolean,* etc. Once all the characteristics are identified, the rainfall example can be defined as such: *rainfall rate for attribute rainfall is 15.00 to 17.00 mm/hr*.

B. Language concepts

In this section, the key language concepts which cover both syntax and semantic related topics will be illustrated. They contain: 1) how to handle the inclusion, exclusion, and conditional ODD statements, 2) how to specify default language state, 3) how to structure an ODD description – including composition statement and conditional statement, 4) the operators used in the language concepts, 5) how to handle extension to the domain model, 6) how to reference and provide versioning for the domain model.

1) Inclusion, exclusion, and conditional

As described earlier, one of the goals of defining the ODD specification is to have the ability to map any given situation to a situation that is in/out of the defined ODD boundary. This results in the requirement for a mechanism which defines inclusion and exclusion for each of the attributes. However, the inclusion and exclusion of certain attributes can have dependencies on other attributes elsewhere in the taxonomy, therefore the language should also have the capability of handling conditional statements. Three keywords are introduced as qualifiers to serve this purpose, as shown in Table II, which are *suitable, unsuitable, conditional*, they are intended to be used for each attribute that needs to be described.

TABLE II. QUALIFIERS FOR INCLUSION, EXCLUSION AND CONDITIONAL

Qualifiers	*Meaning*
Suitable	Included in the ODD
Unsuitable	Excluded from the ODD
Conditional	Inclusion/exclusion have dependencies

Inclusion of an attribute at any level of the hierarchy has implications on the inclusion/exclusion of its sibling attributes. Each ODD statement will tackle one parent class and will perform the classification of its leaf node attributes into the three categories. Upon covering all the parent classes a complete ODD description can be produced. In the ODD language, whether an attribute is described using *Suitable* or *Unsuitable* is determined by the efficiency of writing the statement. For instance, the road type as defined in the BSI PAS 1883 contains *motorways, radial roads, distributor roads, minor roads, slip roads, parking and shared space*. In order to write a statement for the road type such that only *motorways* are suitable, one could state *Suitable road type is motorway*. Alternatively, one could also state *Unsuitable road type are radial roads, distributor roads, minor roads, etc*. Both approaches are syntactically and grammatically correct, however the first one is the more efficient approach. In addition, both approaches can also be used to infer what sibling classes are excluded or included. Understanding of the domain model hierarchical structure is therefore key in writing an efficient ODD description.

Often the inclusion or exclusion of an attribute has dependencies on other conditions. For example, if one wishes to express that motorway is only suitable to drive during sunny condition, in this case the inclusion or exclusion of *motorway* is dependent on the *sunny condition.* The conditional qualifier is introduced to handle such dependencies. It is used to mark any attributes that have dependencies, and in a separate section of the ODD statement the exact dependencies are elaborated in detail, this will be illustrated later.

A potential issue could occur when the underlying domain model is extended to contain additional leaf node attributes within the existing hierarchical branches. For example, considering a previously defined ODD states that *Suitable road type is motorway* (which also mean all the other sibling classes under road type are unsuitable), if the user updates the road type branch to contain a custom road type, then this newly added road type will be automatically excluded. The opposite problem also exists if previously defined ODD uses *Unsuitable*, then the newly added classes will automatically become suitable. A solution to this issue is to include the versioning information of the domain model within the ODD specification, so that the ODD statements and the domain model can be updated synchronously.

2) Default state

Another issue one could encounter so far is that stating the inclusion/exclusion/conditionals for each ODD branch can be inefficient especially when majority of the attributes use the same qualifier (inclusion or exclusion). For example, if a system is only limited by the illumination (e.g., unsuitable under low-ambient condition) but all other ODD attributes are suitable, it would be more efficient to only state the unsuitable attribute rather than providing the suitable/unsuitable for all the attributes. However, only stating the unsuitable attribute would leave all the other attributes undefined in the specification and result in an incomplete ODD. To address this issue, the language provides a mechanism for the user to specify a default state of the ODD specification, which can be *permissive* or *restrictive*. Please note that here the permissive and restrictive are applied as a global condition, it is different from the suitable and unsuitable illustrated earlier, which are

applied to individual statements. Using a permissive default state the assumption is made that attributes which are not mentioned are included in the ODD, while using the restrictive base state the assumption is made that these attributes are excluded from the ODD. This language feature is used to improve the efficiency of the written ODD description, such that every attribute of the taxonomy need not be written about for the description to be considered complete.

In practice, when paired with the description statements that combine permissive and restrictive language, a permissive base state has proven more efficient [18] in describing ADS' that are towards level 4 or higher of the SAE J3016 levels of driving automation [7], or roadways that contain a high amount of attributes and conditions from BSI PAS 1883 [6]. On the other hand, a restrictive base state has proven more efficient when describing ADS' that conform to level 2 or 3 of the levels of driving automation [18].

Similar to the previous section, a potential issue associated with the permissive/restrictive arises when the underlying domain model is extended. When the domain model is updated to contain additional class branches or parent classes for the leaf node attributes, such branch will be automatically included or excluded under the global assumption of permissive and restrictive. This issue can also be resolved by including the versioning information of the domain model.

3) Composition and conditional statement

While a certain number of ODD attributes' constraints might be specified directly as suitable or unsuitable within a specification, the others will have conditions attached and their dependencies will need to be further specified. To clearly differentiate these two categories of attributes, the composition and conditional statements are introduced as parts of the language. Composition statements contain those attributes without any dependencies associated, in addition it also provides indications on the attributes that have dependencies. In the conditional statement, those previously indicated attributes will be defined in detail. Figure 5 displays an example of the structure of an ODD specification. It can be seen that within the composition statement, each of the relevant attributes are associated with qualifiers such as suitable, unsuitable and conditional. In addition, the conditional attribute will be marked by a custom identifier 'Cond1', and its dependencies are then described within the conditional statement section.

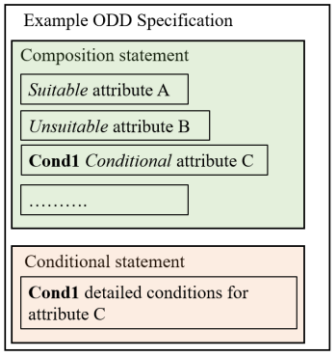

Figure 5: Example structure of an ODD specification

```
# 'Composition' Statement for suitable and unsuitable
Qualifier Current attribute is [Attribute properties]
# 'Composition' Statement for conditional
Custom identifier Qualifier Influenced attribute is
[Influenced attribute properties]

# Example 1
Suitable Road type is [Motorway]
# Example 2
Cond_1 Conditional Road type is [Motorway]
```

Figure 6: Composition statement syntax and examples

Within each individual composition statement, the syntax follows the form displayed in Figure 6. As illustrated, the syntax can be divided into two variations, one for suitable/unsuitable statements and one for conditional statement. For the suitable/unsuitable statements, it contains *qualifier*, *current attribute*, and its *associated properties*. Qualifiers are introduced earlier which can be 'Suitable', 'Unsuitable' or 'Conditional'. The current attribute refers to the attribute that is being defined, when paired with a well-defined domain model, all the parent and sibling classes can be inferred to provide additional semantics. The attribute properties consist of the metrics, the data type and the unit that forms the individual constraints. The syntax for the conditional cases contains *custom identifier, qualifier* and *influenced attribute,* and *attribute properties*. The custom identifiers are used to indicate the condition being described, in the example it is 'Cond1' but it can be under any user defined names. Furthermore, the syntax of the conditional statement also contains the influenced attribute, this is the attribute which has dependencies. Figure 5 also contains two individual examples, example 1 illustrates that *Motorway* is the only suitable *Road type*. In this case, *Motorway* is of enumeration type without unit, and it is considered as the attribute property for *Road type* attribute. Example 2 states that *Motorway* under *Road type* has further dependencies that need to be defined, and such dependency is identified under the name *Cond_1*.

Building on top of the structure of the composition statement, the syntax for the conditional statement contains more complexity. As displayed in Figure 7, the conditional statement contains *custom identifier, qualifier, metric, influencing attribute, influencing attribute properties* and *influenced attribute properties*. The *custom identifier* uses the same identifier names as indicated in the composition statement, its purpose is to reference back to the influenced attributes. The *qualifiers* used in here will be either 'suitable' or 'unsuitable'. The *metric* refers to the intended measuring metric for the influencing attribute, for example *speed* (measuring metric) can be used for *wind* (influencing attribute). The *influencing attribute* refers to the attribute that the *influenced attribute* is dependent on. The attribute properties could include attribute value and unit. An example

```
# 'Conditional' Statement General Structure
Custom identifier Qualifier Attribute Metric of
Influencing attribute for [Influenced attribute
properties] is [Influencing attribute properties]

# Example
Cond_1 Unsuitable Speed of Wind for [Motorway] is
[greater than 30.0 m/s]
```

Figure 7: Conditional statement syntax and examples

is shown in Figure 7, it describes that the suitability of *Motorway* is dependent on the *wind speed,* it would be unsuitable when *wind speed* is *greater than 30.0 m/s. Greater than* in this context is an operator for the language, which will be covered in the next section.

4) Operators

An important element of the ODD language is operators, they are used to define constraints and also provide logical expression consists two or more constraints. For the ODD language the operators are expressed using structured natural language format, they can be divided into mathematical operators and logical operators. Mathematical operators consist: greater than, greater than or equal to, less than, less than or equal to, equal to. The logical operators include: AND, OR, NOT.

TABLE III. OPERATORS USED IN THE ODD LANGUAGE

Mathematical operators	*Logical operators*
greater than (>)	AND
greater than or equal to (≥)	OR
less than (<)	NOT
less than or equal to (≤)	
equal to (=)	

5) Extension mechanism

The domain model used in the ODD language is intended to be extensible, similar extensibility requirement is also highlighted in the BSI PAS 1883 [5], and the base domain model only serves as a starting point. In practice, different users might want to further add custom attributes on top of the base domain model. For example, one might want to add a private test track as a road type, and use it to construct the ODD specification of a system under development. To satisfy this requirement, the ODD language has included an extension mechanism, the idea is that user would take a common base domain model and use the extension mechanism to modify the domain model within the ODD specification, such modifications will then be traceable to other users of the same specification. The custom attributes can be assigned metrics, types, values and units where necessary, and they can be seamlessly integrated with the base domain model. Within the ODD specification structure, such extension is placed before composing the ODD statements.

Figure 8 illustrates the syntax of the extension mechanism together with two examples. It can be seen that it includes *new attribute, parent attribute, metric, data type* and *attribute properties.* Please note that depending on the data type of the new attribute, it might not need the corresponding metrics, values and units, therefore the second statement within the syntax is optional. Example 1 demonstrates how user could add a new attribute called '*my_rain*' under *rainfall* parent class, furthermore *my_rain* is defined using rainfall rate with unit mm/hr and it is constrained using a double range between 0.1 and 2.5. Example 2 illustrates how one could add a new road type called '*my_test_track'* under the *road type* parent class. In this case since *my_test_track* does not require defining metrics, values or data type, the rest of the information is therefore not needed for this extension. One could consider that *road type* is an attribute with enumerated data type, and example 2 shows how the enumerations can be extended.

```
# 'Extension' syntax
- Add New attribute to Parent attribute
(- Add [Metric] with datatype [Attribute properties] to New attribute)

# Example 1
- Add my_rain to Rainfall
- Add [Rainfall rate] with float range [0.1 to 2.5 mm/hr] to my_rain
# Example 2
- Add my_test_track to Road type
```

```
# Domain model versioning syntax
Include: Name of the standards [date of release,
access info]
```

Figure 9: Domain model versioning syntax

```
Include: PAS 1883:2020 Operational Design Domain (ODD) taxonomy
for an automated driving system (ADS) - Specification [2020,
https://www.bsigroup.com/en-GB/CAV/pas-1883/]
Base state: Permissive
Extension: N/A

#Composition statements
Unsuitable Drivable area type is [Minor roads]
Suitable Number of lanes is [greater than 2]
Suitable Lane dimension is [greater than 3.7]
Suitable lane type is [Traffic lane]
Suitable Direction of travel is [Left-hand drive]
Suitable Curve is [less than 1/500 m]
Unsuitable Transverse plane is [Undivided, Barriers on edge]
Suitable Drivable area surface type is [Asphalt, Concrete]
Unsuitable Drivable area signs is [Part-time signs]
Cond_1 Conditional Drivable area type is [Motorway]

#Conditional statements
Cond_1 Unsuitable Weather for [Motorway] is rainfall
```

Figure 10: Case study example converted to ODD language format

6) Domain model reference

While the extension mechanism provides traceability of any modifications made to the domain model, the reference to the domain model itself will also need to be clarified in the first place. The traceability of any modification is only valid when knowing the exact domain model. Therefore, it is compulsory to provide domain model reference within the ODD description under the current ODD language specification. This is an important factor due to the possible safety implications of applying a description to an incorrect taxonomy. This language was developed with the use of BSI PAS 1883 and has been completed using the first version of this standard. Standards by nature are ever changing, updating and going through multiple iterations as the industry progresses. Figure 8 displays the syntax for specifying the version of the domain model, the *name of the standards*, the *date of release* and the relevant *access information* are required to be specified. Such versioning information will be stated at the beginning of an ODD specification.

IV. CASE STUDY

In this case study, the example from BSI PAS 1883 Annex A is used for a comparison between tabular description and the ODD language format. The domain model used in this case study is consistent with BSI PAS 1883 ODD taxonomy, and the tabular format is shown in Table IV. As can be seen, apart from 'yes' and 'no' in the capability column, the table also displays a dependency on the *motorway* attribute - that *motorway* is only suitable when there is no rainfall. The capability of the rest of the attributes contain a mixture of 1) 'yes/no', and 2) 'yes' under certain constraints. The format of

the capability column description is similar to commentary, it is a rather informal way of documenting and is not directly parsable or executable.

Figure 10 illustrates the same ODD specifcation after converting into the ODD language presented in this paper. In addition to the information within the table, the ODD description also contains the reference to the domain model, the base state and extension information. Utilising the permissive defualt state, the main body of the ODD specification displays a much more compact form without sacrificing details. In addition, the dependecy of *motorway* attribute on *weather* has been captured within the conditional statement. Due to the consistency in the language grammar, this ODD description can be easily parsed and integrated with the testing workflow. On the other hand, benefiting from using a structured natural language base, it is also human-readable and can be used by non-technical users.

TABLE IV. TABULAR ODD EXAMPLE

Attribute	***Sub-attribute***	***Sub-attribute***	***Capability***
Drivable area type	Motorways	-	Yes, when no rainfall
	Radial roads		Yes
	Distributor roads		Yes
	Minor roads		No
Lane spec	Number of lanes	-	Yes, minimum of two lanes
	Lane dimensions		Minimum 3.7m
	Lane type	Bus lane	No
		Traffic lane	Yes
		Cycle lane	No
		Tram lane	No
		Emergency lane	No
		Other special purpose lane	No
	Direction of travel	Right-hand traffic	No
		Left-hand traffic	Yes
Drivable area geometry	Horizontal plane	Straight roads	Yes
	-	Curves	Yes – up to 1/500m
	Vertical plane	Up-slope	Yes
		Down-slope	Yes
		Level plane	Yes
	Cross-section	Divided/undivided	Divided
		Pavement	Yes
		Barrier on the edge	No
		Types of lanes together	Traffic lane
Drivable area surface type	Asphalt	-	Yes
	Concrete		Yes
	Cobblestone		No
	Gravel		No
	Granite setts		No
Drivable area signs	Type	Regulatory	Yes
		Warning	Yes
		Information	Yes
	Time of operation	Part-time	No
		Full-time	Yes
	State	Variable	Yes
		Uniform	Yes

V. CONCLUSION

This paper presents a structured natural language based textual format for the description of the ODD. The ODD can be divided into two aspects: the underlying domain model, and the language concepts. The domain model used in this paper has been aligned with ODD standards, it defines the class hierarchy and attributes properties, and it is represented in the form of a domain ontology. The language concepts section, on the other hand, considers both the semantic and syntactic aspects. They cover how to handle the inclusion, exclusion and conditional statement, how to enable extension and versioning of the domain model, how to define the default state. The syntax is intended to be both human and machine-readable. The dedicated, machine readable counterpart of this ODD language concept is covered in part II [19]; by using query semantics to implement the concepts covered in this paper, an executable formal language is established.

I. ACKNOWLEDGEMENT

The work presented in this paper has been carried under the Innovated UK and Centre for Connected and Autonomous Vehicles (CCAV) funded OmniCAV project (Grant No. 104529). This work is also supported by UKRI Future Leaders Fellowship (Grant MR/S035176/1). The authors would like to thank the WMG center of HVM Catapult and WMG, University of Warwick, UK, for providing the necessary infrastructure for conducting this study. WMG hosts one of the seven centers that together comprise the High Value Manufacturing Catapult in the UK.

REFERENCES

[1] N. Balfe, S. Sharples, and J. R. Wilson, "Impact of automation : Measurement of performance , workload and behaviour in a complex control environment," *Appl. Ergon.*, vol. 47, pp. 52–64, 2015, doi: 10.1016/j.apergo.2014.08.002.

[2] S. Le Vine, X. Liu, F. Zheng, and J. Polak, "Automated cars : Queue discharge at signalized intersections with ' Assured-Clear-Distance-Ahead ' driving strategies," *Transp. Res. Part C Emerg. Technol.*, vol. 62, pp. 35–54, 2016, doi: 10.1016/j.trc.2015.11.005.

[3] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings, "Identifying a gap in existing validation methodologies for intelligent automotive systems: Introducing the 3xD simulator," in *Proc. of the IEEE Intelligent Vehicles Symposium 2015*, 2015, pp. 648–653.

[4] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings, "The Science of Testing: An Automotive Perspective," 2018, doi: 10.4271/2018-01-1070.

[5] S. Khastgir, S. Birrell, G. Dhadyalla, and P. Jennings, "Calibrating trust through knowledge: Introducing the concept of informed safety for automation in vehicles," *Transp. Res. Part C Emerg. Technol.*, vol. 96, pp. 290–303, 2018, doi: 10.1016/j.trc.2018.07.001.

[6] "Operational Design Domain (ODD) taxonomy for an automated driving system (ADS) – Specification," *The British Standards Institution, BSI PAS 1883*. 2020.

[7] SAE, "Surface Vehicle Recommended Practice: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016)," 2018. doi: 10.4271/2012-01-0107.

[8] International Organization for Standardization, "Road vehicles — Taxonomy for operational design domain for automated driving systems ISO34504," 2021.

[9] SAE Industry Technologies Consortia, "AVSC Best Practice for Describing an Operational Design Domain: Conceptual Framework and Lexicon," *Avsc00002202004*, p. 26, 2020.

[10] M. Gyllenhammar *et al.*, "Towards an Operational Design Domain

That Supports the Safety Argumentation of an Automated Driving System," *10th Eur. Congr. Embed. Real Time Syst.*, pp. 1–10, 2020.
[11] M. S. Blumenthal, L. Fraade-blanar, R. Best, and J. L. Irwin, *Safe Enough: Approaches to Assessing Acceptable Safety for Automated Vehicles*. .
[12] P. Koopman, B. Osyk, and J. Weast, *Autonomous vehicles meet the physical world: Rss, variability, uncertainty, and proving safety*, vol. 2. Springer International Publishing, 2019.
[13] H. Farah *et al.*, "An Empirical Analysis to Assess the Operational Design Domain of Lane Keeping System Equipped Vehicles Combining Objective and Subjective Risk Measures," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2589–2598, 2020, doi: 10.1109/TITS.2020.2969928.
[14] H. Grönninger, H. Krahn, B. Rumpe, M. Schindler, and S. Völkel, "Textbased Modeling," *Proc. 4th Int. Work. Softw. Lang. Eng. (ateM 2007)*, no. 4, 2007.
[15] E. Thorn, S. Kimmel, and M. Chaka, "A Framework for Automated Driving System Testable Cases and Scenarios - Report No. DOT HS 812 623," 2018.
[16] X. Zhang, S. Khastgir, and P. Jennings, "Scenario Description Language for Automated Driving Systems: A Two Level Abstraction Approach," 2020.
[17] I. Colwell, B. Phan, S. Saleem, R. Salay, and K. Czarnecki, "An Automated Vehicle Safety Concept Based on Runtime Restriction of the Operational Design Domain," *IEEE Intell. Veh. Symp. Proc.*, vol. 2018-June, pp. 1910–1917, 2018, doi: 10.1109/IVS.2018.8500530.
[18] R. Myers, Design considerations for ODD ontology, Executive Summary, CPC, 2020.
[19] E. Schwalb, P. Irvine, X. Zhang, S. Khastgir, P. Jennings, "A Two Level Abstraction ODD Definition Language: Part II", IEEE SMC 2021(Submitted).
[20] S.Khastgir, S. Brewerton, J.Thomas, and P.Jennings, 2021. Systems Approach to Creating Test Scenarios for Automated Driving Systems. Reliability Engineering System Safety, p.107610.