Deliverable D3.6 - "CUDS specification and reference implementation"

Responsible Partner:	UCL	31.08.2023
Contributor(s):	Fraunhofer, SINTEF, UNIBO	13.08.2023
Reviewer(s):	Bjørn Tore Løvfall (SINTEF) Willem van Dorp (UNR)	29.08.2023 31.08.2023
Coordinator:	CMCL Innovations	31.08.2023
Dissemination Level:	Public	
Due Date:	M30 (June, 2023)	
Submission Date:	31 August 2023	

Project Profile

Programme	**Horizon 2020**
Call	**H2020-NMBP-TO-IND-2020-twostage**
Topic	**DT-NMBP-40-2020** **Creating an open marketplace for industrial data (RIA)**
Project number	**953163**
Acronym	**DOME 4.0**
Title	**Digital Open Marketplace Ecosystem 4.0**
Start Date	**December 1st, 2020**
Duration	**48 months**

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953163. It is the property of the DOME 4.0 consortium and do not necessarily reflect the views of the European Commission.

Document History

Version	Date	Author	Remarks
V0.1	19.05.2023	Mike Wang	Initial template
V0.15	**20.05.2023**	**Adham Hashibon**	Adapted Intro
V0.5	13.8.2023	Adham Hashibon	Report on recent CUDS Developments
V0.6	29.8.2023	Adham Hashibon	Address review comments from CMCL and SINTEF
V0.7	31.8.2023	Adham Hashibon	Add lessons learnt

Executive Summary

The existing SimPhoNy Common Universal Data Structures (CUDS, https://github.com/simphony/osp-core, implemented in Python) have been extended and adapted to support multiple features needed for the DOME 4.0 marketplace eco system. The new CUDS are hosted on a new official "future features" branch maintained by the UCL IMD Group (https://github.com/SimPhoNy-Future/osp-core) for continuous development, advancement and distribution of SimPhoNy. The DOME 4.0 CUDS (DOME-CUDS) support efficient semantic data exchange across platforms providing support for granular levels of application specific semantics. The CUDS specification has been extended with simple relations that enable composition of structured data entities and triples including support for Literals and physical units. This will enable readily an implementation for the applications and platforms needed by the showcases. An updated, functional form of the DOME 4.0 eco system and Data Set Ontologies are proposed, enabling the direct use of the ontology in applications. The CUDS are data structures that draw their structure and semantic attributes and properties directly from the underlying ontologies. Support for serialization of the CUDS into JavaScript Object Notation (JSON) and Linked JSON (JSON-LD) constructs is enhanced to enable implementation and handling of data in a distributed web environment. Finally, a lightweight triple store is developed (sigraDB) for CUDS objects that will enable rapid development of the DOME 4.0 clearing house, standardised semantic connector and enhanced brokerage.

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

1. Introduction

The DOME 4.0 web platform is a marketplace of marketplaces dealing with data. It focuses on creating an open data platforms ecosystem that interoperate on various levels. The interoperability in DOME 4.0 is not aimed only at exchange of large datasets, which remains within the realm of each platform, but on making the assets and resources **findable**, through its web based search engine, **accessible**, through its connector delegating operations to the respective platforms, **interoperable**, through use of open standards, and **reusable** (in short FAIR), through the use of open ontology based standards which enable rapid discovery of the data without the need for human intervention. The Common Universal Structures (CUDS) play a major role in providing for the FAIRness aspects as it allows encapsulating the entire semantic (meaning) of the data along with the data itself augmenting it with tools and API for easy handling and management. In this document we report efforts in Task 3.6 aimed at updating the existing SimPhoNy CUDS to the demands of DOME.

We start by a short introduction of existing web-based data structures, and then explain the need for an added semantic value through the integration with ontologies stemming from the OntoCommons community and then introduce how the SimPhoNy CUDS , especially the updated ones (the DOME CUDS), fulfill these needs for DOME 4.0. Finally, we demonstrate the use of CUDS for a simple dataset case.

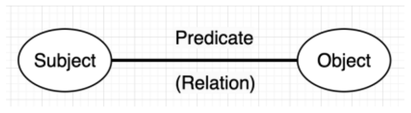


Figure 1: The concept of an RDF Triplet

1.1 Web data standards and Graph data structures

The need to represent complex information and data about various applications, user stories, case scenarios, platforms, and manufacturing or modelling processes calls for advanced data structures that can cover both data and metadata in a connected consistent manner. World Wide Web Consortium (W3C) based technologies such as DCAT[1], SKOS[2], FOAF[3], PROVO[4], etc., offer a way to represents entities and catalogues of data but are developed from the onset with world web applications in mind. They focus primarily on textual representation of resources on the web as Uniform Resource Identifiers (URI) that are a superset of the Uniform Resource Names (URN) which can be e.g., an ISBN of a book or a Uniform Resource Locator (URL) in the form of the well-known html addresses, e.g., 'https://www.example.com/webpage.html`.

[1] World Wide Web Consortium. 2020. "Data Catalog Vocabulary (DCAT) - Version 2." https://www.w3.org/TR/vocab-dcat-2/
[2] World Wide Web Consortium. "SKOS Simple Knowledge Organization System Reference." 2009. https://www.w3.org/TR/skos-reference/
[3] Brickley, Dan, and Libby Miller. "FOAF Vocabulary Specification 0.99." 2014. http://xmlns.com/foaf/spec/
[4] World Wide Web Consortium. "PROV-Overview: An Overview of the PROV Family of Documents." 2013. https://www.w3.org/TR/prov-overview/

They lack the ability and versatility to represent deeply hierarchical large data structures that represent the intertwined hierarchical structure of materials, its related applications and properties. Yet, these W3C based data models are based on profound and widely adopted technology, most notably the Resource Description Framework (RDF) and the Ontology Web Language (OWL). The RDF is a standard for describing any resource on the web as a subject-predicate-object triplet (see Figure 1). The subject denotes a resource which the data or information is *about*, the predicate is the part that states something relating to the subject, and the object is the resource that is relating to the subject. Note that in general an object can itself act as a subject in another triple statement. In this way numerous links between various resources and relations (predicates) can be built to fully provide a complete knowledge-based representation of the data.

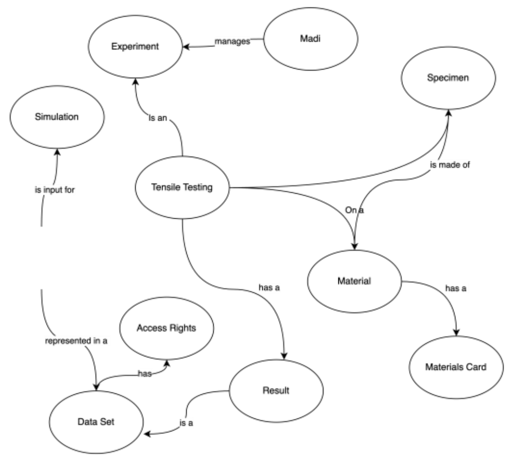


Figure 2: The concept of Graph Data Structure as means to relate concepts and relation to metadata.

The RDF are essentially means to represent **graph data structures** (See Figure 3) with nodes representing entities and edges representing the connections or relations. In fact, when referring to data we often find the need to explain the metadata. For example, given a set of data points representing Temperature as a function of Time in a, say, comma separated values (CSV) file, we need to specify either in the file itself, or as a separate document to be conveyed with the data that the first column is time and the second is temperature (or vice versa). This is in addition to any other information that may be needed to allow a human agent to understand how to interpret and use the data, like units. Graph data structures in general, and RDF in particular, allow us a much more straightforward and robust means not only to convey this metadata to a human agent, but make it interpretable by a machine (i.e., an algorithm). Perhaps most importantly, the metadata and the data can become indistinguishable, hence alleviating the need to maintain them separately. As shown in the Figure 2, instead of a dump of data in rows, a graph enables a transparent representation that is traceable by an algorithm.

In RDF, the nodes (or vertices) of the graph are the resources represented as URIs and the edges or connections (relations) between the nodes being the predicates and represented as either Object Properties (also URIs) relating two resources, or as Data Properties relating a resource entity to a lateral value (e.g. string, or number). One more aspect noteworthy of mentioning in the realm of RDF is the distinction between the terminological foundation, which defines the vocabulary (terminology) representing various resources and their relations from the actual instances of such resources. In other word, an Atom for instance can represent a resource of a term, while Al and O are two different instances (assertions) of the term. The terms (also known as the T-Box) provides the domain of interest through the specification of the vocabulary representing domain concepts and the relations between them. This terminological knowledge can be thought of as defining the types of entities that exist in a domain. The specific instances of domain entities are then specific individuals that belong to one of the types in the T-Box.

The T-Box is normally represented by an ontology, which defines the entities and their relations. While powerful, these web ontologies rely heavily on the textual description of the URI[5] and the existence of a qualified resource (e.g. a web page). In essence, the web linked data structures are designed for connecting disparate resources on the internet which are web pages.

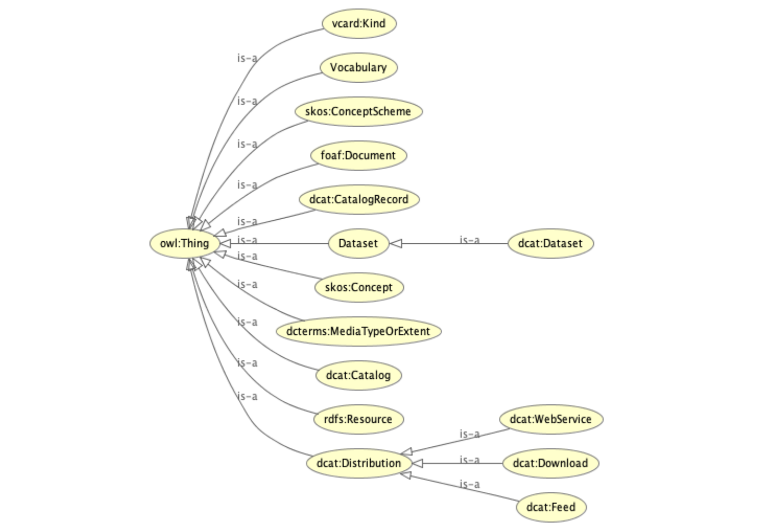


Figure 3: An example of an ontology representing the T-Box of the data Catalouge ontology DCAT which is a W3C standard.

Certain applications, especially from the domain of materials science, engineering and manufacturing require more versatile data structures that represent entities that may or may not have corresponding qualified resources in the W3C sense, i.e. they do not have a web “home pages), and their definition may not be simply given by a URI but rather require more intricate information. Moreover, significant portion of the data and information that needs to be coded may require various conceptual models that do not necessarily have to map to URI’s. Furthermore, these conceptual models may be serialised, or need to be serialised into various formats, e.g., to be fed into various modelling tools or analysis software, which make it difficult to manage.

A more coherent, and domain specific data framework is needed, that on the one hand is compliant with the W3C standard, to achieve widest interoperability with existing web technologies and take advantage of existing tools, and at the same time enables adaptation to the technical needs of the applications at hand.

[5] Berners-Lee, Tim, Roy Fielding, and Larry Masinter. 2005. "Uniform Resource Identifier (URI): Generic Syntax." Request for Comments 3986. Internet Engineering Task Force. January 2005. https://www.ietf.org/rfc/rfc3986.txt.

1.2 The Common Universal Data Structures (CUDS)

The Common Universal Data Structures (CUDS) were developed originally through the SimPhoNy FP7 Project [6,7]. The CUDS can be seen as a programmatic incarnation of ontology into an object-oriented framework of classes (the T-Box) and objects (the A-Box). Each CUDS class conveys, and in fact carries within, the entire axiomatic knowledge that an ontology concept supports, and consequently any CUDS object (data) carries the same information. This allows on the fly inference and logic to be applied on the data, rather than needing to rely on external resources or URIs.

Figure 4 shows schematically how an ontology concept (a class) which has its own data and object (relations) properties as well as restrictions and axioms given as a textual representation (see Table 1) can be converted by the SimPhoNy Open Semantic Platform[8] (OSP) into an object-oriented CUDS class (currently implemented using Python) with data properties casted as class attributes, and the axioms and restrictions as object methods. The properties (i.e., the predicates) are converted to properties that relate one CUDS class (or object) to another. The CUDS Classes can then be instantiated in an object-oriented manner into objects which preserve (inherit) the same attributes, methods, and properties.

The CUDS classes and objects are distinguished from the OWL RDF classes and instances, respectively, in that they are programmatically query-able, including tracing back the full

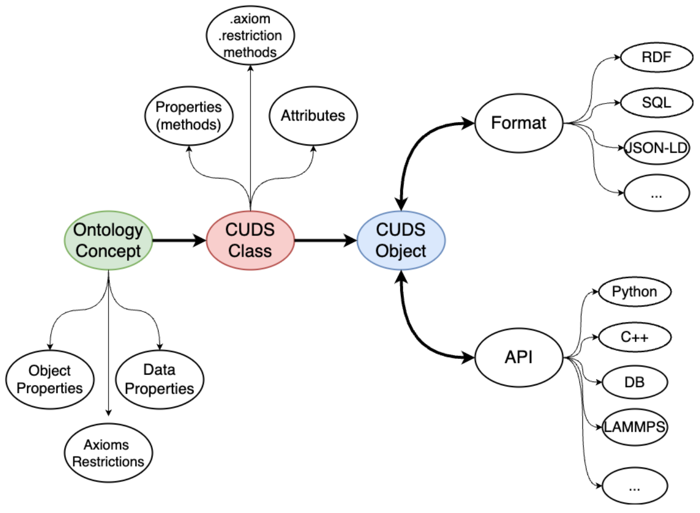


Figure 4: The concept of the CUDS demonstrating their versality transforming and representing an ontology concept or individual in multitute of forms.

[6] SimPhoNy FP7 Project. 2019. "SimPhoNy-OSP." GitHub repository. Accessed August 14, 2023. https://github.com/simphony/simphony-osp.

[7] SimPhoNy Project. 2023. "SimPhoNy Documentation (v4.0.0)." Accessed August 14, 2023. https://simphony.readthedocs.io/en/v4.0.0/.

[8] Note, originally OSP referred to Open Simulation Platform, however, since the development has moved to the UCL IMD Group, the focus is on providing a more general Open Semantic Platform for data management.

inheritance and ontological structure as they carry the entire ontological information within. In the next section this concept will be explained in more details with examples.

1.3 Updated CUDS Specifications

A simple yet powerful extension of the CUDS[9] is performed to enable rapid and efficient representation of data structures. A CUDS entity can be related to another entity through a relation or connection as follows:

```
CudsA.connect(CudsB, rel=ontologyRelation)
```

CUDS can have attributes by assigning a data property axiom on the ontology level. For example, to assign a mass to say, a sample, one would need to define:

```
CUDS.Sample mass some xsd:double
```

Then in SimPhoNy OSP one can create a specific sample and assign a mass like so:

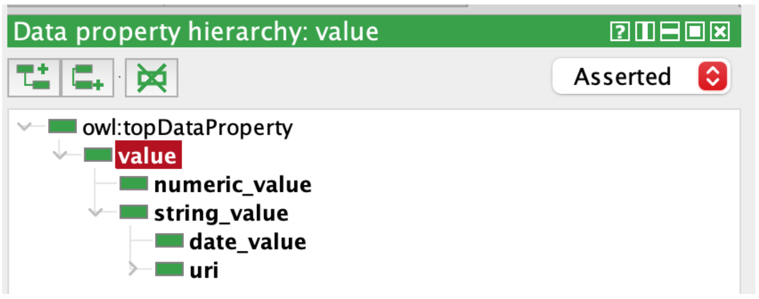
```
my_sample(mass=5)
```

However, one needs to also assign a unit, e.g., like so:

```
my_sample(unit=”Kg”)
```

This approach requires that each property or attribute like mass and unit above to be defined within the ontology as data properties, which will limit their use as classes (or at least create an overhead to maintain them) without any added value.

In the new SimPhoNy-Future, we define a universal data property which is called `value` (Figure 4), that serves as the parent property for all properties. In fact, we limit such properties to the most generic types of Literals rather than specific properties. A universal connection between these valued properties and the respective material, or entity attribute is made via a semantic triple relation:


Data property hierarchy: value

Asserted

- owl:topDataProperty
  - value
    - numeric_value
    - string_value
      - date_value
      - uri

Figure 5: The definition of data properties in CUDS.

[9] SimPhoNy Project. 2023. "SimPhoNy Documentation (v4.0.0)." Accessed August 14, 2023. https://simphony.readthedocs.io/en/v4.0.0/.

```
This_mass – Cuds.Mass(value=5, units = “Kg”)

my_sample.connect(this_mass, rel=has)
```

In this way, the valued (numerical, textual, etc) appears on equal footing as other classes in the ontology and facilitate rapid query.

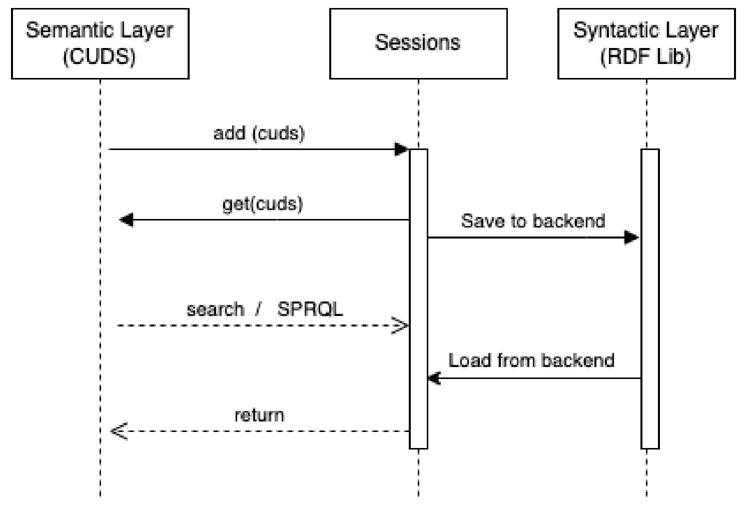


Figure 6: SimPhoNy provides a CUDS interface to Sessions which act as a container (box) of CUDS including relations. The Session connects to multiple backend engines, one of which is an RDF presistent storage.

A second update to the CUDS is done on the relations. In EMMO and other heavy weight ontology, there are multitude of parthood relations, from the `hasPart`, to the more obscure ‘`isConcomitantWith`’. While these may provide added expressive power to the ontology, they can be cumbersome to use for most cases. Moreover, normally the actual relation type is easily inferable on the fly from the actual relation. Hence, in the DOME CUDS we rely on one top relation for all containment or parthood, which is simply a has for a general relation and `hasPart` for topological ones.

These two simple updates are sufficient to enable CUDS in DOME to provide means for defining complex data structures when combined with the SimPhoNy OSP framework for creating and managing the CUDS. In the next chapter a short introduction to OSP is given followed by a full example of a CUDS.

2. The SimPhoNy OSP

The SimPhoNy Open Semantic Platform aims at facilitating seamless semantic interoperability between various tools, platforms, and agents. In the context of DOME 4.0, we are interested in the ability to map CUDS to ontology on the one end, and the connection to RDF enabled backends, see Figure 5 on the other. We shall not provide a complete coverage of the SimPhoNy platform here, and the reader is referred to the official documentation[10]. We focus here on the elements that are required for DOME 4.0. Moreover, we emphasise that while SimPhoNy was developed originally by the FP7 SimPhoNy project consortium, it has since been further developed through numerous EU projects, including MarketPlace, SimDOME, ReaxPro. Recently, the main development of SimPhoNy[11] has moved formally to the Data Driven Group at the UCL Institute for Materials Discovery IMD. The current source code, with all modifications adapting the package for DOME 4.0 are found in the new repository SimPhoNy-Future[12].

Two main features of SimPhoNy-Future OSP are of use for DOME 4.0. The first is the ability to consume existing ontologies, such as DCAT, EMMO, etc., and provide for each class a corresponding python class. Second, the ability to manage the instances of the classes in a semantic backend that supports SPARQL queries, as well as interoperability of the existing W3C eco system supported at least partially by other platforms. Hence, in DOME 4.0 we focus on using an RDF compliant backend engine for providing a persistent storage as well as mapping of CUDS to RDF triplets and back without loss of information.

2.1.1 General Work on SimPhoNy-Future in DOME 4.0

Efforts within Task 3.6 in DOME 4.0 consisted of several bug fixes and feature enhancements including preserving the hierarchy of super classes and support for multiple heterogenous ontologies. The latter is crucial for DOME 4.0 as we need support for linking classes across disparate ontologies stemming from different eco systems and communities. For example, one can link an ontology class `emmo.Dataclass` from the EMMO namespace, with an ontology `domeo.Crystallography` from the dome core ontology, with a relation `dcat.keyword` from the DCAT namespace. Moreover, better support for `rdfschema` and export (serialization) of CUDS into Json and Json-ld. See[13] for a comprehensive list of enhancements done in DOME 4.0.

2.2 A new lightweight DOME 4.0 Graph Database

As the main goal of DOME 4.0 is to enable a semantic backend that connects assets across platforms in a semantic manner, there is a need to enable the storage of RDF and CUDS triplets as well as efficient query of such linked data. To this end, UCL developed a light-weight open-source graph database focused on supporting mainly scientific application data on top of

[10] https://simphony.readthedocs.io/en/v4.0.0/index.html
[11] https://github.com/simphony/simphony-osp
[12] https://github.com/SimPhoNy-Future/osp-core
[13] https://github.com/SimPhoNy-Future/osp-core/issues?q=is%3Aissue+

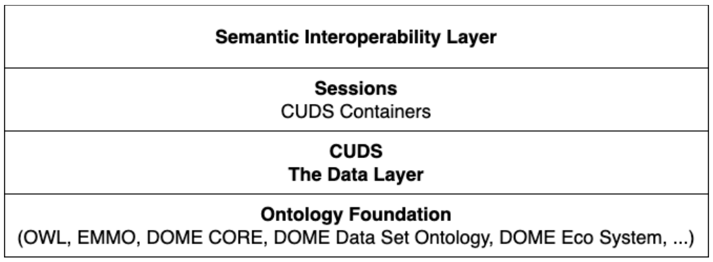

Semantic Interoperability Layer
Sessions
CUDS Containers
CUDS
The Data Layer
Ontology Foundation
(OWL, EMMO, DOME CORE, DOME Data Set Ontology, DOME Eco System, ...)

Figure 7: SimPhoNy-Future OSP takes an ontology and converts it into a CUDS class used in the session to create indivisual instances of data and provides an interoperability layer with simple API for creating, searching (iterating) of CUDS.

SimPhoNy-Future. This lightweight scientific graph database (sigraDB) provides a persistent backend for storing and managing CUDS data structures and is hence a core part of the CUDS eco system. It includes provision for full SPARQL operations. The main architecture is shown in Figure 7. The development of the code can be found here[14]. Currently efforts in DOME 4.0 are directed towards replacing the use of the closed source Allegro Graph with SigraDB starting from the Knowledge service, and then adding new functionality to enable seamless implementation of the Clearing House and Connectors.

[14] https://github.com/SimPhoNy-Future/sigraDB

3. Implementing the DOME 4.0 CUDS Data Structures

The main advantage of utilising SimPhoNy-Future both for the representation of the CUDS and the backend sigraDB is that developing and implementing further extensions of CUDS for DOME boils down to simply providing the proper ontology and installing it into the sigraDB and DOME backends. Instantly, once the ontology is installed, DOME 4.0 can support the full power of CUDS. In the following we provide first an overview of the efforts to streamline the existing DOME 4.0 eco system core ontology for SimPhoNy-Future and then the development of a lightweight DOME 4.0 ontology (DOMEO) that contains the Data Set Ontology developed in Task 3.1. We show then how this enables to define a CUDS for a general dataset, platform, and persona. More examples will be uploaded to the sigraDB repository mentioned before.

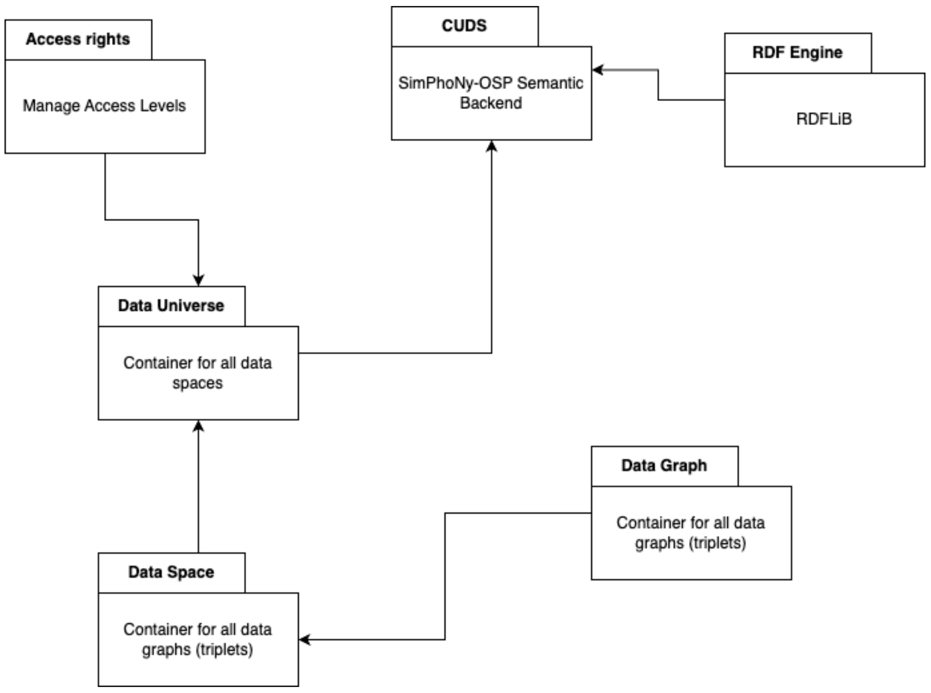

Access rights
Manage Access Levels
CUDS
SimPhoNy-OSP Semantic Backend
RDF Engine
RDFLiB
Data Universe
Container for all data spaces
Data Graph
Container for all data graphs (triplets)
Data Space
Container for all data graphs (triplets)

Figure 8: The various classes of sigraDB showing the data universe which contains a number of data spaces each containing a graph of triplets CUDS. Access to the universe, spaces and CUDS can be controlled per user and group. The SigraDB backend will use a git service to keep full provenance of the changes and enables updates with rollbacks as needed.

3.1 The adaptation of the DOME 4.0 DataSet Ontology

In D3.1 the DOME 4.0 DataSet Ontology was defined by mapping DCAT, FOAF, etc to the emmo classes. While elegant, this work is rather theoretical and not directly practical. Here, we reintroduced the same concepts within a light-weight ontology developed specifically for this purpose and that is designed to also be consistent and integratable with the OntoCommons eco system. In particular it allows for easy augmentation with EMMO. In this ontology, an entity named DomeDataSet (see Figure 9) is defined with parts referring to the basic DCAT based metadata.

This ontology also implements the CUDS updated specifications, namely provides a valued class and simple has and `hasPart` relations. These are particularly useful for defining the parts of the data set as shown in Figure 10. In the Dome Data Set Ontology one can infer the various metadata elements as semantic entities that are part of the dataset description. Notice the definition of the Semantic Tier level in the ontology (Figure 10) which accounts for how much knowledge does DOME 4.0 about a data set and the corresponding access it has. Tier 1 datasets are those that

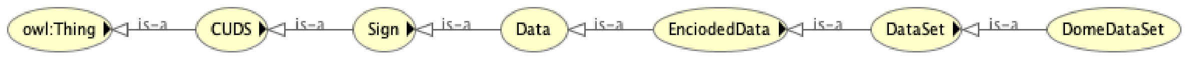


Figure 10: Aligning the DOME 4.0 Data Set Class under a general CUDS class to enable better interoperability.

DOME knows only the set of metadata shown in the figure, i.e., it is consistent with the DCAT element. Tier 2 provides additional information about the data sets that exist internally with respect to an application, while Tier 3 contains the entire knowledge about the data set in a fully semantic manner. These are internal to DOME 4.0 and are used in the brokerage algorithm to match the users to proper data. For example, if the user is seeking to use data coming from a Lammps MD simulation, it may need to know which applications can access the data in its row format (Tier 2).

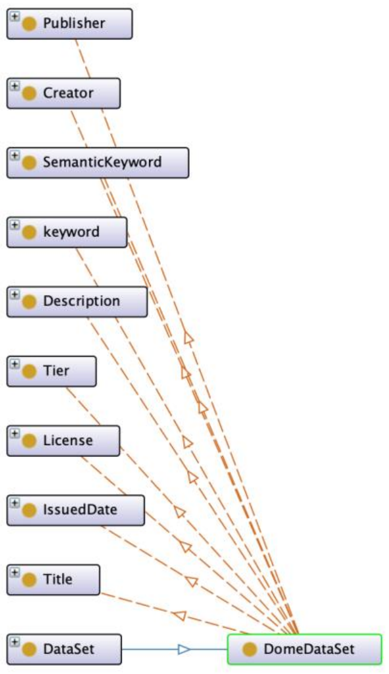


Figure 9: Aligning the DOME 4.0 Data Set Class under a general CUDS class to enable better interoperability.

4. Demonstration of the CUDS for a DomeDataSet

See Appendix 1 for a detailed python example of utilizing CUDS and building it for a DOME dataset.

5. Conclusions / Next steps

We have updated CUDS with simple relations and means to attach values and units to physical properties and developed a new backend persistent storage system in the form of a lightweight graph database. CUDS are therefore serializable into web standard formats (e.g. TTL, OWL, JSON, JSON-LD), stored and managed natively in a graph database and triple store, are directly connected to ontology classes. The CUDS are able to describe native DOME 4.0 Data Sets and resources in a semantic manner.

In the next steps, a more standardised connector service and provenance system will be implemented that adds additional semantics to the already existing semantic systems. The new implementation will enable better easier maintenance for future connectors and provenance systems. The semantic ontology-based backend of CUDS enables rapid development of such components.

6. Lessons learnt

Significant progress has been made in DOME 4.0 so far, developing a semantic Broker, Connector and full provenance system required deeper integration with ontology. At the same time, utilising commercial graph databases proved to be cumbersome to maintain and introduced additional dependencies while DOME 4.0 did not utilise most of the functionality provided being a meta-marketplace. Furthermore, the initial Data Set ontology developed was too theoretical (or idealistic) to be implemented practically into the system and we relied on imposing development guidelines to use the ontology terms and compliant schemes.

The current developments allow to remedy these shortcomings and enable more rapid development. An important lesson pertains then to the need to have practicable and actionable ontology that directly brings an impact on the development. At times, one needs to compromise, at least temporarily on the desire for a complete, philosophically, and logically consistent ontology and focus on the practical aspects that enable direct implementation. The use of CUDS that directly relies on and requires an ontology underlying it, enforced this notion in essence. CUDS necessitates that ontology is actionable and practicable rather than being left as guidelines for developers. Nonetheless, in DOME 4.0 the commitment for ontology as a semantic basis is strong and the use of CUDS makes it easier to use the ontology directly. It opens additional avenues for python-based reasoning enabling to build algorithms that adapt to the ontology enhancing therefore the experience of the users as well. Moreover, the integration of the sigraDB graph system relieves DOME 4.0 from the need to rely on proprietary licensed closed source software which is a risk factor (as the license may change and we may not have access to the system in the future). sigraDB allows use of various open source graph backends including DOME 4.0 native one just developed or utilising example a APACHE Jena [1]and Fuseki graph DB[2]and SPARQL front end.

Achieving these new developments, of updating the ontology, making it more practicable, developing from scratch a new graph DB addressing the needs for DOME 4.0 and integrating the updated CUDS required significant efforts. Our focus in DOME 4.0 has been from day one to advance in small steps taking into account an Agile approach. This necessitated starting off with the best tools available (e.g. using Allegro Graph) and then switching to the more semantic ones as they become available. We have achieved this albeit with the cost of delay to this deliverable, which was necessary to allow for the entire code to be developed.

7. Deviations from Annex 1

No deviations, other than a two-month delay in submitting the report. The delay did not have an effect of other reports or milestones in the project.

8. References

[1] Hashibon, A., et al. "Common universal data structures (CUDS) and vocabulary in the SimPhoNy integrated framework." (2015).

[2] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, Martin, Vrgo c, Domagoj. (2016). Foundations of JSON schema. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 263–273). See also https://en.wikipedia.org/wiki/JSON

[3] Gregg Kellogg, Pierre-Antoine Champin, Dave Longley. JSON-LD 1.1 – A JSON-based Serialization for Linked Data (W3C Working Draft). [Technical Report] W3C. 2019. hal-02141614v1

[4] Sporny, Manu, et al. "JSON-LD 1.1." *W3C Recommendation, Jul* (2020).

9. Acknowledgement

The author(s) would like to thank the partners in the project for their valuable comments on previous drafts and for performing the review.

Project partners:

#	Type	Partner	Partner full name
1	SME	CMCL	Computational Modelling Cambridge Limited
2	Research	FHG	Fraunhofer Gesellschaft zur Förderung der Angewandten Forschung E.V.
3	Research	INTRA	Intrasoft International SA
4	University	UNIBO	Alma Mater Studiorum – Universita di Bologna
5	University	EPFL	Ecole Polytechnique Federale de Lausanne
6	Research	UKRI	United Kingdom Research and Innovation
7	Large Industry	SISW	Siemens Industry Software NV
8	Large Industry	BOSCH	Robert Bosch GmbH
9	SME	UNR	Uniresearch B.V.
10	Research	SINTEF	SINTEF AS
11	SME	CNT	Cambridge Nanomaterials Technology LTD
12	University	UCL	University College London

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 953163. It is the property of the DOME 4.0 consortium and do not necessarily reflect the views of the European Commission.

Annex 1

To include the python CUDS case found at: https://github.com/SimPhoNy-Future/osp-core/blob/fix-15/examples/future/CUDS_DOME40_SPARQL_1.ipynb