DCSO: Towards an Ontology for Machine-actionable Data Management Plans

João Cardoso (joao.m.f.cardoso@tecnico.ulisboa.pt)
Universidade de Lisboa, Instituto Superior Técnico

Leyla J. Castro
ZB MED Information Center for Lice Sciences

Fajar J. Ekaputra
TU Wien

Marie C. Jacquemot
DMP OPIDoR

Marek Suchánek
Faculty of Information Technology, Czech Technical University in Prague

Tomasz Miksa
TU Wien

José Borbinha
Universidade de Lisboa, Instituto Superior Técnico

Research Article

Keywords: Data Management Plan, Machine-Actionable Data Management Plan, Ontology, Semantic Web Technologies

Posted Date: March 23rd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1458035/v1

DATABASE

DCSO: Towards an Ontology for Machine-actionable Data Management Plans

João Cardoso[1,2*], Leyla J. Castro[3], Fajar J. Ekaputra[4], Marie C. Jacquemot[6,7], Marek Suchánek[8], Tomasz Miksa[4,5] and José Borbinha[1,2]

*Correspondence:
joao.m.f.cardoso@tecnico.ulisboa.pt
[1]Universidade de Lisboa, Instituto Superior Técnico, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal
[2]INESC-ID, R. Alves Redol, 9, 1000-029 Lisboa, Portugal
Full list of author information is available at the end of the article

Abstract

The concept of Data Management Plan (DMP) has emerged as a fundamental tool to help researchers through the systematical management of data. The Research Data Alliance DMP Common Standard (DCS) working group developed a core set of universal concepts characterising a DMP in the pursuit of producing a DMP as a machine-actionable information artefact, i.e.,machine-actionable Data Management Plan (maDMP). The technology-agnostic approach of the current maDMP specification: (i) does not explicitly link to related data models or ontologies, (ii) has no standardised way to describe controlled vocabularies, and (iii) is extensible, but has no clear mechanism to distinguish between the core specification and its extensions. Currently, the maDMP specification provides a JSON serialisation and schema to operationalise the approach. Such approach, however, does not address the concerns above. This paper reports on the community effort to create the DMP Common Standard Ontology (DCSO) as a serialisation of the DCS core concepts, with a particular focus on a detailed description of the components of the ontology. Our initial result shows that the proposed DCSO can become a suitable candidate for a reference serialisation of the DMP Common Standard.

Keywords: Data Management Plan; Machine-Actionable Data Management Plan; Ontology; Semantic Web Technologies

1 Introduction

With the continuous growing of research data and the ultimate goal of sharing FAIR data, researchers face the challenge of systematically managing that data and its corresponding metadata. Data Management Plans (DMP) make it easier for researchers to respond to this challenge. The DMP, produced as a text-based document, describes techniques, methods and policies on how data is produced and managed throughout its life cycle [1]. Additionally, it establishes associations between data management activities and actors responsible for their execution [2].

The concept of DMP as a document has evolved towards that of the machine-actionable DMP (maDMP). The implementation of maDMPs will allow to overcome some of the identified obstacles linked to the current text-based representation of DMPs [3]. One of the main issues is the level of details provided by a DMP, which can vary according to the design choices, awareness, and knowledge of its creators. In addition, having information expressed in free-form text often leads to DMPs with incomplete, inadequate, ambiguous or missing information. The main goal of maDMPs is to have DMPs represented in a format that makes its information

readable and reusable by both humans and automated systems. This machine-actionable representation will allow the exchange of information between automated systems, its integration into existing data management workflows, and will support automated machine-process pipelines regarding data management policies [4, 5]. As a consequence, it will lessen the administrative burden for researchers and facilitate the involvement and support from data management experts and services. Furthermore, it will ease the updating process of DMPs, data management follow-ups, and will provide a linked inventory of the research outputs, research objects, actors, and infrastructures. As a result, DMPs, being machine-actionable, will become a real lever for the implementation of the FAIR principles, thus contributing to "Turning FAIR into reality" [6].

The production of a machine-actionable representation for a DMP implies the necessity for standardisation. To tackle that issue, the Research Data Alliance (RDA)[1] DMP Common Standards (DCS) working group created a set of universal elements characterising a DMP [7, 8, 9, 10], which resulted in the creation of the DCS application profile. The next step of the DCS working group was to provide reference serialisations of this application profile. At the time of release, the DCS application profile was accompanied by a JSON serialisation[2]; The responsibility of developing other serialisation formats lies with the community. In order to enable machines to not only read but also be able to interpret the data, and increase the interoperability between the different services required throughout the research process, there is a need to add a semantic layer on top of this syntactic layer.

The definition of the DCS concepts and its JavaScript Object Notation (JSON) serialisation, however, still leaves a number of open challenges. Among others, these challenges are: (1) the lack of explicit linking to existing ontologies; (2) the lack of mechanisms to describe controlled vocabularies; and (3) the lack of a mechanism that allowed for the extension of DCS set of terms.

This paper reports on the creation of the DMP Common Standard ontology (DCSO), a community effort towards the creation of an ontology serialisation of the DCS application profile. The creation of DCSO aimed to address the aforementioned challenges with the current DCS terms and existing serialisations.

The remaining of this paper details the resulting ontology and its components, and is organised as follows. Section 2 offers a report on the conceptualisation and creation process of DCSO. In particular, it addresses the requirements and methodology followed to achieve this goal. Section 2.2 describes the first attempt at creating a semantic-based serialisation of the DCS application profile. Section 2.3 presents the first stable version of DCSO and its three stages of development. Section 3 presents the adoption of DCSO in the Data Stewardship Wizard (DSW) DMP creation tool, as a use case for DCSO. Finally, Section 4 provides a summarised review on the contents of this paper, as well as a description of the future goals for DCSO.

[1]The Research Data Alliance Website. https://www.rd-aliance.org/rda-europe. Accessed 09 Mar 2022.

[2]The DCS application profile JSON serialisation, in GitHub. https://github.com/RDA-DMP-Common/RDA-DMP-Common-Standard/tree/master/examples/JSON. Accessed 09 Mar 2022

2 Background, Requirements and Methodology

This section will provide an insight into the background, identified requirements, as well as the methodology used to iteratively develop DCSO from its initial version to the latest version. DCSO final aim is to be adopted by the DCS working group as the official serialisation of the DCS application profile.

2.1 The application profile as starting point

To comply with the goal of establishing a set of universal terms to characterise a DMP, the DCS working group has strived to create an application profile. According to the definition, an application profile is a metadata design specification that uses a selection of terms from multiple metadata vocabularies, with added constraints, to meet application-specific requirements[3]. However, due to the fact that not all of the terms selected by the DCS working group are yet associated with established metadata vocabularies, the concept of application profile is not fully materialised, and therefore it is still an ongoing task. Regardless of this fact, and as part of the overall goal, there was the need to develop serialisations of the application profile. These would allow any tools or systems engaged in research data processing, not only to consume data but also to add data to maDMPs, thus automating data interchange.

The DCSO was created as a community initiative with the overall goal of expanding the set of existing serialisations of the DCS application profile, by adding a semantic-based serialisation. With DCSO, information from the application profile is represented using semantic technologies, specifically ontologies, which allow for the representation of a shared conceptualisation of knowledge through the usage of formal semantics [11]. One of the key characteristics behind the selection of ontologies was their extensibility, as concepts can be matched or relations established between ontologies covering distinct domains. This characteristic enforces the suitability of ontologies as a means to represent the DCS application profile, for it is also designed with modularity in mind. Additionally, ontologies enable reasoning, and thereby knowledge inference from the information explicitly represented [12]. In spite of the traditional perception of ontologies as highly formal means of knowledge representation, their suitability for creation of Linked Open Data (LOD) has been proven [13, 14]. The usage of semantic technologies is therefore in compliance with the overarching requirements established by the concept of maDMP.

In the process of creating a semantic-based serialisation of the DCS application profile, three key requirements that DCSO should accomplish were identified. These are: (1) DCSO should allow for ontologies referenced in the DCS application profile to be integrated through the reuse of its terms; (2) DCSO should allow and enforce the usage of controlled vocabularies; and (3) DCSO should be extendable, so as to comply with any future extensions of the DCS application profile. The following sections describe DCSO creation process, from its origins to its current iteration.

2.2 Initial Version

The first attempt at creating a semantic-based serialisation of the DCS application profile took place in the spring of 2019. This version was created to serve primarily

[3]Definition of application profile, according to Dublin Core. https://www.dublincore.org/resources/glossary/application_profile/. Accessed 09 Mar 2022.

as a proof of concept. Thus, the creation process was expedited and simplified by not fully abiding by the best practices in ontology engineering. This first version proved the viability of a semantic-based representation of the DCS application profile, however, it also failed in fully complying with the three key requirements that DCSO concept should accomplish.

The DCS application profile references multiple terms and fields from standardised vocabularies (e.g., Data Catalog Vocabulary (DCAT)[4] and Dublin Core (DC)[5]). In this initial version of DCSO, all of the terms and fields were redefined. This was contrary to the best practice of reusing terms through the integration of existing domain ontologies. In spite of having all terms and fields represented, the lack of reuse of referenced terms and fields meant that this version of DCSO would not meet one of the key requirements for its development.

In order to simplify the creation process, it was decided that a set of custom literal datatypes was to be created to accommodate the usage of the controlled vocabularies specified in the DCS application profile. This solution is in breach of the best practices of ontology engineering, and albeit users of DCSO could potentially use controlled vocabularies, this was not a scalable solution moving forward. In addition to this, the decision was made to represent multiplicity and type constraints through the usage of OWL constraints. Despite this being a viable means of constraint representation in ontology engineering, this solution would not allow for compliance validation of the data with the specification. As a result it was, for all effective purposes, impossible to enforce the use of controlled vocabularies, thus failing to comply with yet another key requirement.

Finally, there were also other issues in the pursuit of following the best practices in ontology engineering, such as using the digital repository links as the ontology namespace, opposed to assigning a persistent namespace Uniform Resource Identifier (URI).

2.3 First Stable Version

With the concept of having a semantic representation of the DCS application profile being proven viable by the first version of DCSO. It was therefore necessary to create a stable version, that would not only comply with the three key requirements for DCSO, but also follow the best practices in ontology engineering. The RDA Hackathon on Machine-Actionable Data Management Plans[15] proved to be the perfect opportunity to tackle this objective. The motivation for the Hackathon was to promote the usage of the maDMP concept by the research community. Participants were encouraged to submit topics and assemble teams that would collaborate for two days to tackle the submitted topics. In line with the motivation of the hackathon, it was decided that the creation of a stable of DCSO should be one of the proposed topics in the hackathon.

The new version of DCSO was to be a fresh start, that would benefit from the experience acquired during the creation the first version. As such, it should comply

[4]Data Catalog Vocabulary - Version 2. https://www.w3.org/TR/vocab-dcat-2/. Accessed 09 Mar 2022.

[5]The Dublin Core specification. https://www.dublincore.org/specifications/dublin-core/dcmi-terms/. Accessed 09 Mar 2022.

with all of the previously identified key requirements, whilst following the best practices in ontology engineering. In order to achieve this objective, it was decided that DCSO was to be organised into DCSO Core and DCSO Extensions (DCSX) (as can be seen in Figure 1). The first would be a representation of the DCS application profile, which would reuse terms from a selection of domain ontologies, whilst the later would support DCSO core by providing an aggregation of all the controlled vocabularies referenced in the DCS application profile. Due to the time constraints imposed by the duration of the hackathon the development process was divided into three iterative stages.

2.3.1 First Stage - DCSO Core

The first stage focused solely on the development DCSO core. The resulting ontology was expressed using the Terse Resource Description Framework (RDF) Triple syntax (Turtle)[6], and reused classes and terms from imported domain ontologies.

In DCSO Core all relations between DCS application profile concepts are represented as object properties, whilst data properties are used to represent DCS application profile terms, as can be seen in Figure 1a. The object properties are named after the class to which they pertain, using a CamelCase notation, and using the prefix '`has`'. This solution solved an existing issue with the DCS application profile, where relations between concepts were left unnamed, with only information regarding their multiplicity being provided. The data properties followed a similar naming convention, with the distinction being that no prefix was added. However, some of the terms in the DCS application profile required compliance with controlled vocabularies. The solution to represent this select set of terms was to use object properties establishing a relation between classes of DCSO Core and classes of DCSX (see section 2.3.2). These object properties followed the same naming convention as those representing relations between concepts in the DCS application profile, e.g., `dcso:hasCurrencyCode`.

2.3.2 Second Stage - DCSX and Validation Layer

The second stage had two objectives: (1) The incorporation of controlled vocabularies in DCSO; and (2) the creation of a constraint validation layer. This led to the development of DCSX and DCSO constraints validation layer, respectively.

The DCS application profile specifies a set of three controlled vocabularies: (1) ISO 639-3 [16], whose language codes are used to represent the language in which multiple concepts (e.g., *dataset*, *distribution*, and *metadata*) are expressed; (2) ISO 3166-1, whose country codes are used to describe the geographical location of where data is hosted; and (3) ISO 4217 [17], whose currency codes are used to identify the currency in which costs associated with data management are being described in the DMP.

In the initial version of DCSO controlled vocabularies were represented using custom literal datatypes. This solution originally implemented solely as a means of simplifying the creation process, was inadequate for a stable version. The hackathon team opted to resort to the creation of a separate ontology that would serve as an extension to DCSO, and where controlled vocabularies could be represented by classes

[6]The Turtle syntax. https://www.w3.org/TR/turtle/. Accessed 09 Mar 2022.

and their terms as individual instances of those classes. The result was the creation of DCSX, and its classes (*dcsx:Language*, *dcsx:Country* and *dcsx:CurrencyCode*), each associated with a set of individual instances, as can be seen in Figure 1b. Additionally, the necessary object properties required to establish relations to classes belonging to DCSO Core were also created.

The motivation for the creation of DCSO constraints validation layer, was to provide users with the means to assess the compliance of their maDMPs with the DCS application profile, whilst also promoting data completeness and consistency. In the initial version of DCSO constraints were represented using the OWL language, however due to the limitations of OWL it was impossible to validate the compliance of individual DMP instances with DCSO. A solution to this issue was to select a constraint representation language that allowed for compliance validation.

Three validation languages are generally regarded as most prevalent: (1) JSON schema; (2) Shape Expression (ShEx) [18]; and the Shapes Constraint Language (SHACL) [19]. None of the three validation languages stands out in terms of dedicated usage in semantic data validation scenarios. As such, the ultimate selection of ShEx as the validation language was not solely based on its adequacy for the task at hand but also on the expertise and familiarity of the team members with the ShEx language. ShEx is a data modelling language used to describe RDF graphs, Sets of individual ShEx expressions are collected into a ShEx schema, defining conditions on element relations, their cardinality (e.g., one or more, zero or more, zero or one, etc.) and their existence (e.g., mandatory or optional).

DCSO constraint validation layer comprises two distinct ShEx schemas, that follow the constraints established in the DCS application profile. The first schema named '`dcso-dmp`', focuses on the validation of the DMP document. As such, it comprises of elements targeting identifiers, contacts, contributors, costs and projects. The second schema named '`dcso-dataset`', focuses solely on the validation of the datasets referenced in the DMP document. This modularity improves readability (for humans) and makes extensions easier to create. Within a schema, one shape per DCSO class is provided with and initial validation of data properties (e.g., dates or strings) and then a validation of relations expressed by object properties. Validation elements related to the *dmp* DCSO class is shown in Figure 2. An explained excerpt corresponding to these shapes is shown in Figure 3.

There are different options to try and test ShEx schemas, here we use the RDFShape validation service supported by the Web Semantics Oviedo Research Group [20, 21]; an early version of this service is also used in the Validating RDF Data online book [22]. RDFShape provides an end-user interface where users can directly enter an instance to be validated (either as direct input, by URL or by file) as well as the corresponding ShEx schema. Here we describe the process step by step using the ShEx schema for DMPs, also illustrated in Figure 4.

1 Go to DCSO validation directory[7] in GitHub;
2 On a different browser window or tab, go to the RDFShape validator service[8];

[7]The DCSO Validation directory, in Github. https://github.com/RDA-DMP-Common/RDA-DMP-Common-Standard/tree/master/ontologies/validation. Accessed 09 Mar 2022.
[8]The RDFShape validator. https://rdfshape.weso.es/shExValidate. Accessed 09 Mar 2022.

3 On Github, copy the content of a valid DMP example from the file $valid - example_dcso.3.0.2.ttl$;
4 Paste that valid example on the first text area of the RDFShape validation service, (e.g., RDF input);
5 On GitHub, copy the content of the DCSO ShEx schema from the file $dcso - dmp.3.0.2.shex$;
6 Paste that schema on the second text area of the RDFShape validation service, (e.g., ShEx);
7 Define what entity should be validated by adding the text $ex : dmp_1@ < dmp >$ to the third text area in the validator, (e.g., $ShapeMap$);
8 Run the validation by clicking the button $Validate$.
9 You should get a passing validation

2.3.3 Third Stage - Human Readability and Dissemination

By the third stage of the development process most of the functional requirements of DCSO had already been met. The objective was therefore to finalise the development process by addressing requirements that would facilitate its use and adoption. To that effect three tasks were considered: (1) The creation of human-readable descriptions both in DCSO Core as well as DCSX. This was done through the usage of *rdfs:comment* descriptions in all created classes, data properties and object properties; (2) The definition of a persistent namespace for DCSO. The lack of a stable URL to act as DCSO namepsace was one of the issues identified in the initial version of DCSO. To address that issue, and comply with the best practices of having the namespace URI being a persistent identifier, the team resorted to the W3ID[9] service of the W3C Permanent Identifier Community Group[10], and registered the identifier '`https://w3id.org/dcso`'; and (3) Revising and expanding the existing documentation on the various artefacts that comprise DCSO. Existing documentation consisted mainly of markup files that were deposited, along with DCSO, in a GitHub repository

2.4 Towards DCSO Adoption by the DCS Working Group

Upon the completion of the first stable version of DCSO, there was a move for its adoption by the DCS working group, as the official serialisation of the DCS application profile. There were however a number of pre-requisites that needed to be met before any proposal for adoption could be formalised, which resulted in a new version of DCSO. The pre-requisites were as follows:

Integration of DCSX namespaces into the ShEX Validation layer. The use of DCSX as an extension of DCSO was rooted on the necessity of having the means to represent controlled vocabularies that applied to a specific set of DCS application profile terms (see Section 2.3.2). However, in order to have a closer representation of the DCS application profile, where the terms covered by DCSX are represented with string values, it was decided that DCSX approach should be

[9]The W3ID service webpage. https://w3id.org/. Accessed 09 Mar 2022.
[10]The W3C Permanent Identifier Community Group website. http://www.w3.org/community/perma-id/. Accessed 09 Mar 2022.

abandoned in favour of having these terms represented as data properties in DCSO Core. As a result, the controlled vocabularies were represented as constraints in the ShEX validation layer, so as to maintain the validation feature idealised in the original vision of DCSO.

Full coverage of the DCS application profile by DCSO A fundamental prerequisite was to ensure that all of the DCS application profile terms were covered by a matching term in DCSO. The methodology adopted to comply with this prerequisite was to perform a direct comparative analysis between the DCS application profile terms and the first stable version of DCSO (see Section 2.3). As a result of this analysis, multiple discrepancies were found as can be seen in Table 1. These discrepancies fell under two categories:

DCS Application Profile		DCSO	
Concept	Term	Term	Type
	created		Data Property
dmp	language	dcso:hasLanguage	Object Property
	modified		Data Property
dmp_id	identifier		Data Property
funder_id	identifier		Data Property
grant_id	identifier		Data Property
host	geo_location	dcso:hasGeoLocation	Object Property
metadata	language	dcso:hasLanguage	Object Property
metadata_standard_id	identifier		Data Property
contact_id	identifier		Data Property
contributor_id	identifier		Data Property
cost	currency_code	dcso:hasCurrencyCode	Object Property
dataset	language	dcso:hasLanguage	Object Property
dataset_id	identifier		Data Property

Table 1: Discrepancies found as a result of the comparative analysis between the DCS application profile and the first stable version of DCSO.

Missing terms These DCS application profile terms were found not to have a corresponding match in DCSO. The most common case was the lack of representation for the various *identifier* terms that are associated with multiple DCS application profile concepts. In the first stable version of DCSO there was an explicit attempt at defining a representation for the DCS application profile identifier concepts, through the *dcso:Id* class, its subclasses (e.g. *dcso:ContactId*, *dcso:ContributorId*, etc.), and the *dcso:identifierType* data property that had the *dcso:Id* class as its domain. However, there was no data property to represent the *identifier* term. The solution to this issue was the creation of the *dcso:identifier* data property, which had as domain the *dcso:Id* class. Two other additional data properties were also created, namely the *dsco:created* and *dcso:modified*.

Terms represented with the wrong type These discrepancies were a direct result of the decision to abandon DCSX as an approach to represent DCS application profile terms that specified a set of controlled vocabularies. The solution to this discrepancy was to change the representation type of DCSO terms from object property to data property. As a result the *dcso:hasLanguage*, *dcso:hasGeoLocation*, and *dcso:hasCurrencyCode* DCSO object properties were

replaced by the *dcso:language*, *dcso:geoLocation*, and *dcso:currencyCode* DSCO data properties.

Adoption of DCS application profile name and versioning scheme. The current name and versions of DCSO are based on the development of the ontology, without any explicit connection with the reference DCS application profile. In order to ensure consistent DCS application profile usage across its various serialisations, the adoption of the 'maDMP' term as the semantic serialisation name, in opposition to DCSO, and the adoption of the DCS application profile version number are under consideration. Such a decision would entail that new versions of the ontology would be released along with any revision of the DCS application profile.

Mechanisms for transformation between JSON and RDF representations of maDMP. Due to the popularity of the JSON serialisation of DCS application profile, it is essential for the adoption of DCSO to provide mechanisms for transforming data between JSON and RDF serialisations. To this end, the availability of JavaScript Object Notation for Linked Data (JSON-LD) context to transform maDMP JSON to RDF serialisation and JSON-LD serialiser to transform maDMP RDF to JSON serialisation are important. We provides an initial approach to execute these functions and provides examples of the transformations[11].

Exemplifications of funder profile as ontology extensions. One of the main advantage of having RDF serialisation of maDMP is its capability to capture data model extensions without having to change the original DCS application profile. To this end, we plan in the future to provide a set of procedures to define such extension to the DCS application profile and evaluate it to define funder profile as ontology extensions.

3 Case Study and Discussion

This Section provides report on the adoption of DCSO by the Data Stewardship Wizard, one of the most popular DMP creation tools currently available for public usage.

3.1 Use Case: Data Stewardship Wizard

Data Stewardship Wizard[12] or DSW, is a tool used for data management planning widely used in the European life-sciences Infrastructure for biological Information (ELIXIR) and beyond [23]. Its versatility has been proven in several customisations such as VODAN-in-a-Box solution as a data-entry tool for case report forms [24] or as a FIP Wizard to allow efficient capturing of FAIR Implementation Profiles [25]. The core idea of DSW is not to copy the structure of any funders DMP template and ask the same (open text) questions. Instead, DSW has a concept of knowledge models, extensible templates for smart questionnaires, where guidance is done in

[11]The Transformation between DMP Common Standard serialisations application, in GitHub. https://github.com/fekaputra/dcso-json. Accessed 09 Mar 2022.
[12]The Data Stewardship Wizzard website. https://ds-wizard.org. Accessed 09 Mar 2022.

multiple but natural ways – explanations, advice, options for answering, follow-up questions, references, and integrations with application programming interfaces (APIs) to provide answer suggestions. With such a questionnaire, one can get an actual document by selecting the desired export template, e.g., Horizon 2020 DMP template [26]. The export templates are done in Jinja2 templating language[13]; therefore, it can produce any textual format and transform questionnaire replies with any limitations.

During the RDA maDMP Hackathon 2020, a new export template was developed. First, defining the mapping between the core DSW knowledge model and the DCS application profile JSON schema was needed. Some of the information was not covered by that moment, and thus new questions were added. Also, several of the questions were adjusted or moved; however, DSW provides a migration mechanism so the users can easily upgrade to a newer version without losing all the replies. The Jinja2 template for maDMP in JSON is straightforward. It basically queries the replies using known universally unique identifiers (UUIDs) of the questions and created an object according to the JSON schema. Then it just plainly pretty-prints the object as a JSON. An export template in DSW may provide several formats. For example, one can export Horizon 2020 DMP documents in the .pdf, .docx, .html, .tex, or .md file formats. DSW similarly supports also RDF export for maDMPs by using DCSO. There is a Jinja2 template for transforming the same object used for JSON to synthesise an RDF file in the Turtle format. It traverses the object (its fields, arrays, nested objects) and outputs the RDF triples according to DCSO. In the first version, the template produced valid RDF but with the use of blank nodes. It turned out that it causes problems with several other tools after being exported from DSW, i.e., lays obstacles in interoperability. The recent version is free of blank nodes by giving every node a unique identifier.

The identifiers of nodes in RDF are of two types. First, there are those concretely defined by JSON schema (and DCSO), e.g., `dcso:DMPId` or `dcso:FunderId`. It is verified if such a user-entered identifier is URI directly or after some transformation (for instance, we may add `https://doi.org/` before a DOI identifier). In the case of a valid URI, it is used as an identifier for the corresponding node. Then, if it is not a valid URI or the entity does not have an ID defined by DCSO, it still needs a URI. It is synthesised using related URI and question/reply UUID, e.g., the URI of the DMP, the URI of the questionnaire and the UUID of dataset reply item. It is also important to point out the integrations in DSW. A user can, for example, select a funder using integration with CrossRef[14], similarly a license from Wikidata[15] or affiliation from the Research Organization Registry (ROR)[16]. The URI of entry is then saved as a part of the reply and here used in the template for corresponding RDF triple following the linked data principles.

Finally, Turtle is not the only RDF export format that DSW supports for maDMPs and DCSO. Using rdflib[17] for automatic transformations allows for export in

[13]The Jinja project website. https://jinja.palletsprojects.com. Accessed 09 Mar 2022.
[14]The CrossRef website. https://www.crossref.org. Accessed 09 Mar 2022.
[15]The Wikidata website. https://www.wikidata.org. Accessed 09 Mar 2022.
[16]The Research Organization Registry Community website. https://ror.org. Accessed 09 Mar 2022.
[17]The rdflib specification. https://rdflib.readthedocs.io. Accessed 09 Mar 2022.

different formats such as RDF/XML, TRiG, N3, or JSON-LD. The export template is developed as open-source[18], and anyone can easily contribute or use it. This approach allows both simple versioning of the template (e.g., when a new version of DCSO is released) and adopting various future extensions to DCSO in independent branches.

3.2 Discussion

The semantic representation of DMPs will facilitate exchange of information between the different services involved in data management as it becomes independent of their technical implementation. The use of standards to represent and annotate the DMP content makes it FAIR. In this ontology, classes are linked to existing and widely-used ontologies such as `DCAT` and `Friend of a Friend (FOAF)`. Therefore, DMP content can be easily linked to other graphs such as PID graphs or Research object graphs. DCSO extension already provides a set of controlled vocabularies for languages, countries and currencies to annotate DMP content. This semantic representation of maDMP allows custom extensions of the format by adding properties to the classes or deriving subclasses from the maDMP classes. Thus it will be possible to add domain- or service-related specificities to the format while conserving the link to this ontology.

4 Conclusions

This paper report on DCSO, the creation process and path towards adoption by the DCS working group as a serialisation of the DCS application profile, as well as its adoption by the Data Stewardship Wizard (DSW), one of most the popular DMP creation tools. DCSO is a semantic representation of the DCS application profile, which cover all terms from the DCS application profile whilst also reusing terms from established domain ontologies. Furthermore, DCSO is equipped with a validation layer, which consist of a set of ShEx constraints that allows DMPs represented through DCSO to be validated against the DCS application profile.

The team in charge of the development and maintenance of DCSO strives to follow the best practices in ontology engineering, whilst continuously trying to update and upgrade DCSO and its components. Currently the team is focusing on preparing a formal proposal for the adoption of DCSO by the DCS working group as an official serialisation of the DCS application profile. Additionally it's also tackling or planning to tackle the following issues: (1) continued pursue of the integration of terms from other established ontologies into DCSO (e.g. the Data Integration for Grant Ontology (DINGO)[19]), thus enriching the DCS application profile and potentially includes controlled vocabularies associated to all of the relevant concepts; (2) Performing semantic validation of DMP documents represented using DCSO would be an ambitious but useful feature, in particular for any funding agency stakeholders. Services aiming to perform the task through the usage of DCSO are currently being considered for the creation of proof of concept tools.

[18]The maDMP template for DSW, in GitHub. https://github.com/ds-wizard/madmp-template. Accessed 09 Mar 2022.

[19]The Data Integration for Grant Ontology specification. https://dcodings.github.io/DINGO. Accessed 09 Mar 2022.

List of abbreviations

API Application Programming Interface
DC Dublin Core
DCAT Data Catalog Vocabulary
DCS DMP Common Standard
DCSO DMP Common Standard Ontology
DCSX DCSO Extensions
DINGO Data Integration for Grant Ontology
DMP Data Management Plans
DSW Data Stewardship Wizard
ELIXIR European life-sciences Infrastructure for biological Information
FOAF Friend of a Friend
JSON JavaScript Object Notation
JSON-LD JavaScript Object Notation for Linked Data
maDMP machine-actionable DMP
RDA Research Data Alliance
RDF Resource Description Framework
ROR Research Organization Registry
SHACL Shapes Constraint Language
ShEx Shape Expression
URI Uniform Resource Identifier
UUID Universally Unique Identifier
XML Extensible Markup Language

Ethics approval and consent to participate
Not applicable

Consent for publication
Not applicable

Availability of data and materials
The datasets generated and/or analysed during the current study are available in the DMP Common Standards working group GitHub repository repository,
https://github.com/RDA-DMP-Common/RDA-DMP-Common-Standard/tree/master/ontologies

Competing interests
The authors declare that they have no competing interests

Funding
This work was supported by the Austrian Research Promotion Agency FFG under grant 877389 (OBARIS). Development and operation of DSW is supported from the ELIXIR CZ research infrastructure project (Ministry of Education, Youth and Sports grant No. LM2018131). SBA Research SBA-K1 is a COMET Centre within the framework of COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna; COMET is managed by FFG. The researchers from INESC-ID were supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UIDB/50021/2020, by project DRE, Imprensa Nacional - Casa da Moeda S.A. (INCM), and Portugal 2020. Finally, the authors would also like to thank the RDA for establishing the community that made this collaboration possible.

Author's contributions
JB, TM and MJ contributed in writing and revising the the manuscript. MS contributed as a software developer, for as a member of the DSW core team he was responsible for the integration of the DCSO in the DSW. He also contributed with writing and revising the manuscript, with specific focus on Section 3. FE was heavily involved in the design and creation of the DCSO, particularly in the creation of its first stable version. FE was also responsible for the development of a mechanism for the transformation between JSON and RDF representations of maDMP. FE was also a contributor in writing and revising the manuscript. LC contributed in the creation of the first stable version of the DCSO, primarily through the creation of its validation layer. LC also contributed in writing and revising the manuscript. Finally JC was the main contributor in the writing and revising of the manuscript and was heavily involved in the creation of all the existing versions of the DCSO.

Acknowledgements

Not applicable

Author details

[1]Universidade de Lisboa, Instituto Superior Técnico, Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal. [2]INESC-ID, R. Alves Redol, 9, 1000-029 Lisboa, Portugal. [3]ZB MED Information Centre for Life Sciences, Gleueler Straße, 60, 50931 Cologne, Germany. [4]TU Wien, Karlsplatz, 13, 1040 Vienna, Austria. [5]SBA Research, Floragasse, 7, 1040 Vienna, Austria. [6]DMP OPIDoR, R. Jean Zay, 2, 54519 Vandoeuvre-lès-Nancy, France. [7]Inist-CNRS, R. Jean Zay, 2, 54500 Vandoeuvre-lès-Nancy, France. [8]Faculty of Information Technology, Czech Technical University in Prague, Thákurova, 9, 160 00 Prague, Czechia.

References

1. Surkis A, Read K. Research data management. Journal of the Medical Library Association: JMLA. 2015;103(3):154.
2. Michener WK. Ten simple rules for creating a good data management plan. PLoS computational biology. 2015;11(10):e1004525.
3. Simms S, Jones S, Mietchen D, Miksa T. Machine-actionable data management plans (maDMPs). Research Ideas and Outcomes. 2017;3:e13086. doi.org/10.3897/rio.3.e13086.
4. Miksa T, Simms S, Mietchen D, Jones S. Ten Principles for Machine-actionable Data Management Plans. PLoS computational biology. 2019;15(3):e1006750. doi.org/10.1371/journal.pcbi.1006750.
5. Miksa T, Oblasser S, Rauber A. Automating Research Data Management Using Machine-Actionable Data Management Plans. ACM Transactions on Management Information Systems. 2021 Dec;13(2). doi.org/10.1145/3490396.
6. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific data. 2016;3(1):1–9.
7. Miksa T, Neish P, Walk P, Rauber A. Defining requirements for machine-actionable Data Management Plans. 2018;http://ifs.tuwien.ac.at/~miksa/papers/2018-iPres-maDMPs.pdf.
8. Miksa T, Cardoso J, Borbinha J. Framing the scope of the common data model for machine-actionable Data Management Plans. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE; 2018. p. 2733–2742.
9. Miksa T, Walk P, Neish P. RDA DMP Common Standard for Machine-actionable Data Management Plans. 2019;doi.org/10.15497/rda00039.
10. Miksa T, Walk P, Neish P, Oblasser S, Murray H, Renner T, et al. Application Profile for Machine-Actionable Data Management Plans. Data Science Journal. 2021 Oct;20(1):32. doi.org/10.5334/dsj-2021-032.
11. Studer R, Benjamins VR, Fensel D, et al. Knowledge engineering: principles and methods. Data and knowledge engineering. 1998;25(1):161–198.
12. Lenzerini M, Milano D, Poggi A. Ontology representation & reasoning. Universit di Roma La Sapienza, Roma, Italy, Tech Rep NoE InterOp (IST-508011). 2004;.
13. Allemang D, Hendler J. Semantic web for the working ontologist: effective modeling in RDFS and OWL. Elsevier; 2011.
14. Jain P, Hitzler P, Sheth AP, Verma K, Yeh PZ. Ontology alignment for linked open data. In: International semantic web conference. Springer; 2010. p. 402–417.
15. Cardoso J, Castro LJ, Miksa T. Interconnecting Systems Using Machine-Actionable Data Management Plans – Hackathon Report. Data Science Journal. 2021;20. doi.org/10.5334/dsj-2021-035.
16. International Organization for Standardization. ISO639-3:2007 Codes for the representation of names of languages — Part 3: Alpha-3 code for comprehensive coverage of languages. International Organization for Standardization; 2007.
17. International Organization for Standardization. ISO 4217:2015 odes for the representation of currencies. International Organization for Standardization; 2015.
18. Baker T, Prud'hommeaux E. Shape Expressions (ShEx) Primer; 2019. https://shexspec.github.io/primer/.
19. Knublauch H, Kontokostas D. Shapes constraint language (SHACL). W3C Candidate Recommendation. 2017;11(8).
20. Labra-Gayo JE. RDFShape: An RDF playground based on Shapes. 2018;p. 4.
21. Labra Gayo JE, Ulibarri Toledo E, Menéndez Suárez P. Shaclex/RDFShape — Playground for RDF, ShEx, SHACL and more;. http://shaclex.herokuapp.com.
22. Labra Gayo JE, Prud'hommeaux E, Boneva I, Kontokostas D. Validating RDF Data. Synthesis Lectures on the Semantic Web: Theory and Technology. 2017 Sep;7(1):1–328. doi.org/10.2200/S00786ED1V01Y201707WBE016.
23. Pergl R, Hooft RWW, Suchánek M, Knaisl V, Slifka J. "Data Stewardship Wizard": A Tool Bringing Together Researchers, Data Stewards, and Data Experts around Data Management Planning. Data Sci J. 2019;18:59. doi.org/10.5334/dsj-2019-059.
24. Suchánek M, Basajja M. VODAN in a Box: Proof of Concept. Zenodo; 2020. doi.org/10.5281/zenodo.4321626.
25. Schultes E, Magagna B, Hettne KM, Pergl R, Suchánek M, Kuhn T. Reusable FAIR Implementation Profiles as Accelerators of FAIR Convergence. In: International Conference on Conceptual Modeling. Springer; 2020. p. 138–147.
26. Koumoulos EP, Sebastiani M, Romanos N, Kalogerini M, Charitidis C. Data Management Plan template for H2020 projects. Zenodo; 2019. doi.org/10.5281/zenodo.2635768.

Figures

Figures

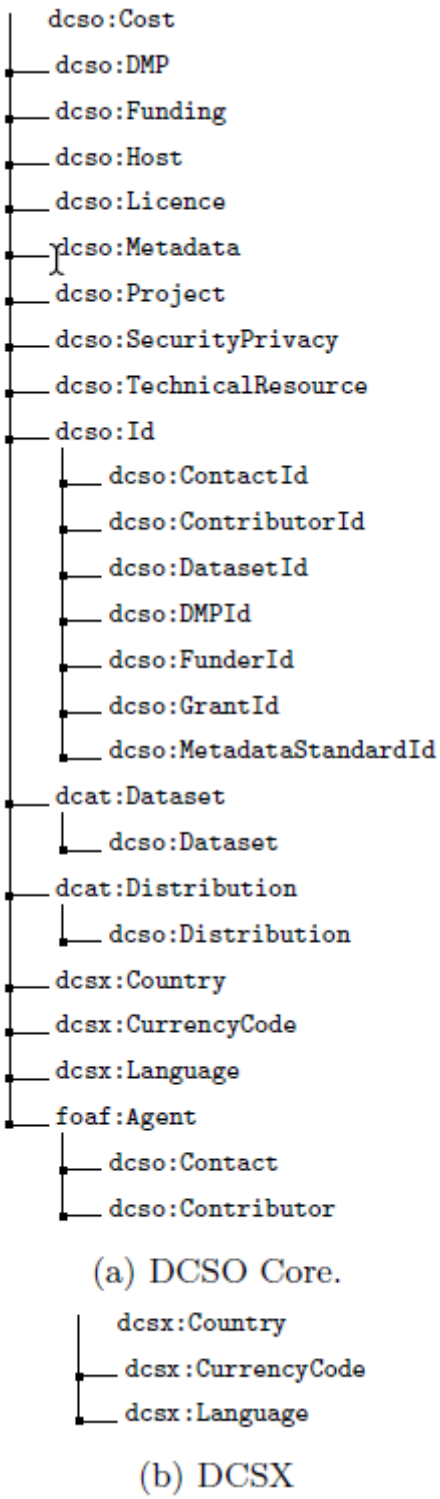

Figure 1

The class structure of DCSO.

a The class structure of the DCSO Core.

b The class structure of the DCSX.

Figure 2

Diagram showing the shapes describing a DMP according to DCSO together with additional elements describing the associated project, cost, contributors and contacts. Validation related to Datasets is not included in this diagram.

Figure 3

Excerpt of a ShEx schema validating a DMP.

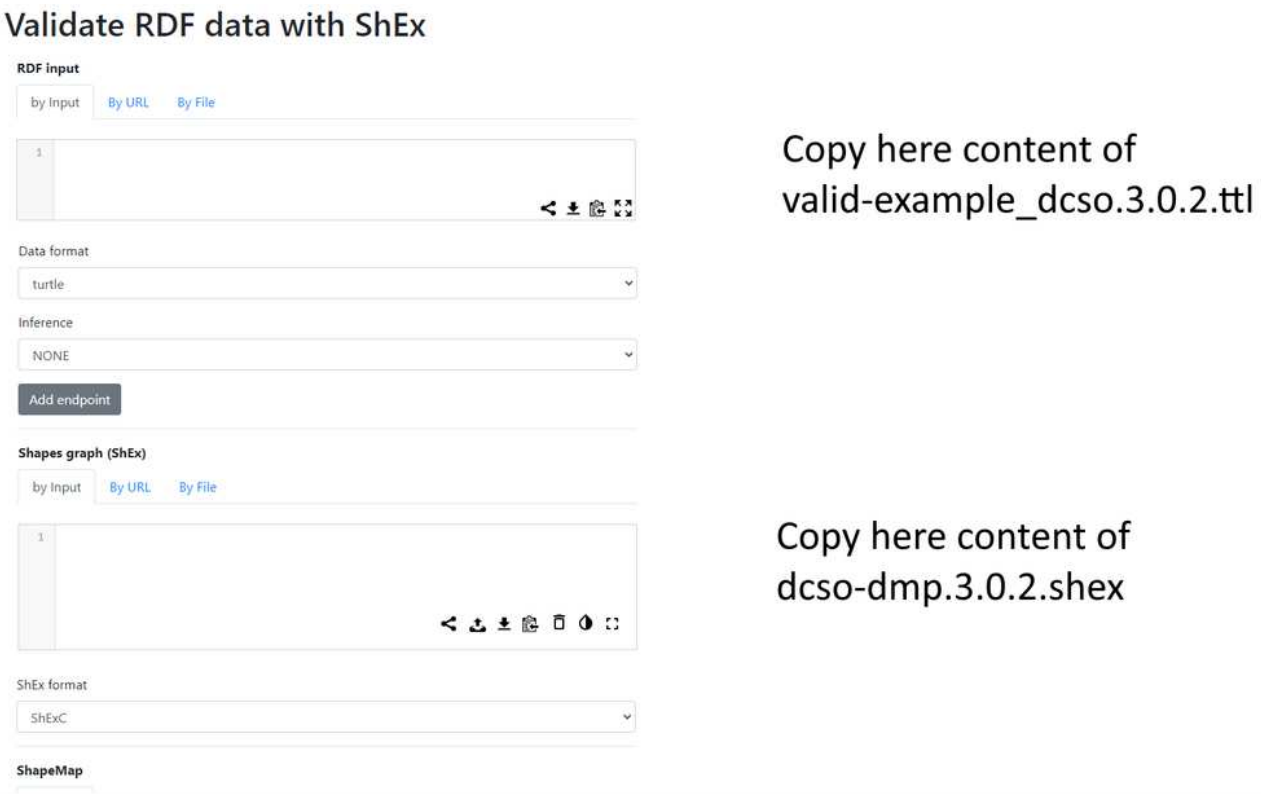

Figure 4

Example of a validation using RDFShape validation service.