Packaging research artefacts with RO-Crate

Stian Soiland-Reyes[??,??,*], Peter Sefton[??], Mercè Crosas[??], Leyla Jael Castro[??], Frederik Coppens[??], José M. Fernández[??], Daniel Garijo[??], Marco La Rosa[??], Simone Leo[??], Eoghan Ó Carragáin[??], Marc Portier[??], Ana Trisovic[??], RO-Crate Community[??], Paul Groth[??], Carole Goble[??]

[a] *Computer Science, The University of Manchester, UK*
[b] *Informatics Institute, University of Amsterdam, The Netherlands*
E-mail: soiland-reyes@manchester.ac.uk; ORCID: https://orcid.org/0000-0001-9842-9718
[c] *Faculty of Science, University Technology Sydney, Australia*
E-mail: Peter.Sefton@uts.edu.au; ORCID: https://orcid.org/0000-0002-3545-944X
[d] *University College Cork, Ireland*
ORCID: https://orcid.org/0000-0001-8131-2150
[e] *Institute for Quantitative Social Science, Harvard University, Cambridge, MA, USA*
ORCID: https://orcid.org/0000-0003-1304-1939
[f] *ZB MED Information Centre for Life Sciences, Cologne, Germany*
ORCID: https://orcid.org/0000-0003-3986-0510
[g] *VIB-UGent Center for Plant Systems Biology, Gent, Belgium*
ORCID: https://orcid.org/0000-0001-6565-5145
[h] *Barcelona Supercomputing Center, Barcelona, Spain*
ORCID: https://orcid.org/0000-0002-4806-5140
[i] *Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain*
ORCID: https://orcid.org/0000-0003-0454-7145
[j] *PARADISEC, Melbourne, Australia*
ORCID: https://orcid.org/0000-0001-5383-6993
[k] *Center for Advanced Studies, Research, and Development in Sardinia (CRS4), Pula (CA), Italy*
ORCID: https://orcid.org/0000-0001-8271-5429
[l] *Vlaams Instituut voor de Zee, Oostende, Belgium*
ORCID: https://orcid.org/0000-0002-9648-6484
[m] *Institute for Quantitative Social Science, Harvard University, Cambridge, MA, USA*
ORCID: https://orcid.org/0000-0003-1991-0533
[n] *https://www.researchobject.org/ro-crate/community (see appendix B)*
[o] *Informatics Institute, University of Amsterdam, The Netherlands*
E-mail: p.t.groth@uva.nl; ORCID: https://orcid.org/0000-0003-0183-6910
[p] *Computer Science, The University of Manchester, UK*
E-mail: carole.goble@manchester.ac.uk; ORCID: https://orcid.org/0000-0003-1219-2137

2451-8484 © 2021 – IOS Press. All rights reserved.

Abstract. An increasing amount of researchers support reproducibility by including pointers to and descriptions of datasets, software and methods in their publications. However, scientific articles may be ambiguous, incomplete and difficult to process by automated systems. In this paper we introduce RO-Crate, an open, community-driven, and lightweight approach to packaging research artefacts along with their metadata in a machine readable manner. Based on Schema.org annotations in JSON-LD, RO-Crate aims to establish best practices to formally describe metadata in an accessible and practical way for their use in a wide variety of situations.

An RO-Crate is a structured archive of all the items that contributed to a research outcome, including their identifiers, provenance, relations and annotations. As a general purpose packaging framework for data and their metadata, RO-Crate is used across multiple areas, including bioinformatics, digital humanities and regulatory sciences. By applying ”just enough” Linked Data standards, RO-Crate simplifies the process of making research outputs FAIR while also contributes to enhance research reproducibility.

Keywords: Data publishing, Data packaging, FAIR, Linked Data, Metadata, Reproducibility, Research Object

1. Introduction

The move towards open science has increased the need and demand for the publication of artefacts of the research process [**?**]. This is particularly apparent in domains that rely on computational experiments; for example, the publication of software, datasets and records of the dependencies that such experiments rely on [**?**].

It is often argued that the publication of these assets, and specifically software [**?**], workflows [**?**] and data, should follow the FAIR principles [**?**]; namely, that they are Findable, Accessible, Interoperable and Reusable. These principles are agnostic to the *implementation* strategy needed to comply with them. Hence, there has been an increasing amount of work in the development of platforms and specifications that aim to fulfil these goals [**?**]. Important examples include data publication with rich metadata (e.g. Zenodo [**?**]), domain-specific data deposition (e.g., PDB [**?**]) and following practices for reproducible research software [**?**] (e.g. use of containers).

These strategies are focused primarily on one *type* of artefact. To address this, [**?**] introduced the notion of **research objects** – *semantically rich aggregations of (potentially distributed) resources that provide a layer of structure on top of information delivered in a machine-readable format*. A Research Object combines the ability to bundle multiple types of artefacts together, such as CSV files, code, examples, and figures. This provides a compelling vision as an approach for implementing FAIR. However, existing research object implementations require a large technology stack, are tailored to a particular platform and are also not easily usable by end-users.

To address this gap, a new community came together [**?**] to develop **RO-Crate** - an *approach to package and aggregate research artefacts with their metadata and relationships*. The aim of this paper is to introduce RO-Crate and assess it as a strategy for making multiple types of research artefacts FAIR. Specifically, the contributions of this paper are as follows:

1. an introduction to RO-Crate, its purpose and context;
2. a guide to the RO-Crate community and tooling;
3. and an exemplar usage of RO-Crate for different artefacts in different communities as well as its use as a connective tissue for such artefacts.

The rest of this paper is organised as follows. We first describe RO-Crate, the assumptions underlying it, and define RO-Crate technically and formally. We then proceed to introduce the community

[*]Corresponding author. E-mail: soiland-reyes@manchester.ac.uk.

and tooling. We move to analyse RO-Crate with respect to usage in a diverse set of domains. Finally, we present related work and conclude with some remarks including RO-Crate highlights and future work.

2. RO-Crate

RO-Crate provides a lightweight approach to packaging research artefacts with their metadata. To illustrate this, let us imagine a research paper reporting on the sequence analysis of proteins obtained from an experiment on mice. The sequence output files, sequence analysis code, resulting data and reports summarising statistical measures or outputs are all important and inter-related research outputs, and consequently would ideally all be co-located in a directory and accompanied with their corresponding metadata. In reality, some of the artefacts (e.g. data or software) will be recorded as external references, not necessarily captured in a FAIR way. This directory, along with the relationships between its constituent digital artefacts, is what the RO-Crate model aims to represent, linking together all the elements pertaining to an experiment and required for its reproducibility.

The question then arises as to how the directory with all this material should be packaged in a manner that is accessible and usable by others. By usable we mean not just readable by humans but programmatically accessible. A de facto approach to sharing collections of resources is through compressed archives (e.g. a zip file). This solves the problem of “packaging”, but it does not guarantee downstream access to all artefacts in a programmatic fashion, or the role of each file in that particular research. This leads to the need for explicit metadata about the contents of the folder, describing each and linking them together.

Examples of metadata descriptions across a wide range of domains[1] abound within the literature, both in research data management (?cite) and within library and information systems (?cite). However, many of these approaches require knowledge of metadata schemas, particular annotation systems, or the use of obscure or complex software stacks. Indeed, particularly within research, these requirements have led to a lack of adoption and growing frustration with current tooling and specifications [**?**].

RO-Crate seeks to address this complexity by:

1. being easy to understand and conceptually simple;
2. providing a strong and opinionated guide regarding current best practices;
3. adopting de-facto standards that are widely used on the Web.

In the following sections we show how the RO-Crate specification and ecosystem achieves these goals, which concur in forming our definition of “lightweight”.

2.1. Conceptual Definition

A key premise of RO-Crate is the existence of a wide variety of resources on the Web that can help describe research. As such, RO-Crate relies on concepts from the Web, in particular that of Linked Data [**?**]. Figure **??** shows the main conceptual elements involved in an RO-Crate; an RO-Metadata File(top) describes the research object using structured metadata including external references, coupled with the contained artefacts (bottom) bundled and described by the RO-Crate.

fig:conceptual width=”90

[1]<https://rdamsc.bath.ac.uk/scheme-index>

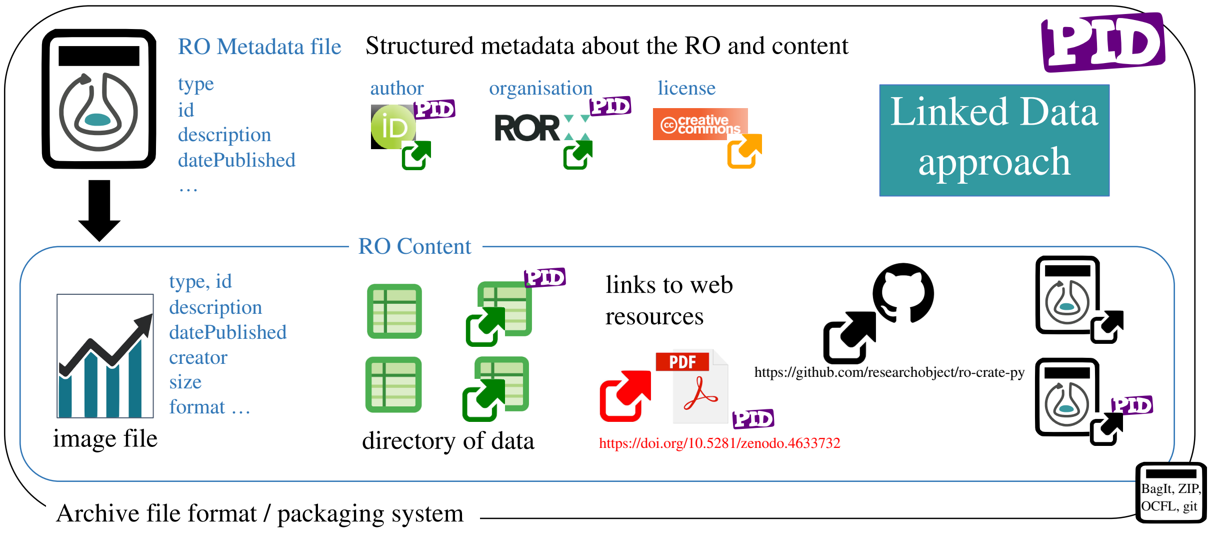

Fig. 1. **Conceptual overview of RO-Crate**. A *Persistent Identifier* (PID) [@doi:10.1371/journal.pbio.2001414] identifies a *Research Object* (RO), which is archived using BagIt [**?**], OCFL [**?**], git or ZIP. The RO is described within a *RO Metadata File*, providing identifiers for authors using ORCID, organisations using [ROR](https://ror.org/) and licences such as Creative Commons. The *RO-Crate content* is further described with its own metadata. Data can be embedded files and directories, as well as links to external web resources, PIDs [@10.1371/journal.pbio.2001414] and nested RO-Crates.

2.1.1. *Linked Data as a core principle*

Linked Data principles [**?**] (use of IRIs to identify resources (i.e. artefacts), resolvable via HTTP, enriched with metadata and linked to each other) are core to RO-Crate; therefore IRIs[2] are used to identify an RO-Crate, its constituent parts and metadata descriptions, and the properties and classes used in the metadata.

Linked Data make it possible for consumers to follow links for more (ideally both human- and machine-readable) information; as the RO-Crate relies on these principles, it can be sufficiently *self-describing* as artefacts can be interrelated using global identifiers, without needing to recursively fully describe every referenced artefact.

[PROPOSED REPHRASE ON PREVIOUS PAR] RO Crates are *self-described*; and follow the Linked Data principles to describe all of their resources in both human and machine readable manner. Hence, resources are identified using global identifiers; and relationships between two resources are defined with links.

The foundation on Linked Data and shared vocabularies also means multiple RO-Crates and other Linked Data resources can be indexed, combined, queried, integrated or transformed using existing Semantic Web technologies such as SPARQL[3] and knowledge graph triple stores.

2.1.2. *RO-Crate is a self-described container*

An RO-Crate is defined as a self-described **Root Data Entity** that describes and contains *data entities*, which are further described using *contextual entities*. A **data entity** is either a *file* (i.e. a set

[2]IRIs[**?**] are a generalisation of URIs (which include well-known http/https URLs), permitting international Unicode characters without %-encoding, commonly used on the browser address bar and in HTML5.

[3]<https://www.w3.org/TR/sparql11-overview/>

of bytes stored on disk somewhere) or a *directory* (i.e. dataset of named files and other directories) — while a **contextual entity** exists outside the information system (e.g. a Person, a workflow language) and it is defined by its metadata. The representation of a **data entity** as a set of bytes makes it possible to store a variety of research artefacts including not only data but also, for instance, software and text.

The Root Data Entity is a directory, the RO-Crate root, identified by the presence of the **RO-Crate Metadata File** (`ro-crate-metadata.json`). This is a JSON-LD file that describes the RO-Crate, its content and related metadata using Linked Data. JSON-LD is an RDF serialisation that has become popular as it is easy to read by humans while also offers some advantages for data exchange on the Internet (e.g. it removes some of the cross-domain restrictions that can be present with XML). JSON-LD is the preferred and widely supported format by RO-Crate tools and community.

The minimal requirements for the root data entity metadata[4] are `name`, `description` and `datePublished`, as well as a contextual entity identifying its `license` — additional metadata are frequently added depending on the purpose of the particular RO-Crate.

RO-Crate can be stored, transferred or published in multiple ways, such as BagIt [**?**], Oxford Common File Layout [**?**] (OCFL), downloadable ZIP archives in Zenodo or through dedicated online repositories, as well as published directly on the Web, e.g. using GitHub Pages. Combined with Linked Data identifiers, this caters for a diverse set of storage and access requirements across different scientific domains, from metagenomics workflows producing hundreds of gigabytes of genome data to cultural heritage records with access restrictions for personally identifiable data.

2.1.3. Data Entities are described using Contextual Entities

RO-Crate distinguishes between data and contextual entities[5] in a similar way to HTTP terminology's early attempt to separate *information* (data) and *non-information* (contextual) resources [**?**]. Data entities are usually files and directories located by relative IRI references within the RO-Crate Root, but they can also be Web or restricted resources identified with absolute IRIs.

As both types of entities are identified by IRIs, their distinction is allowed to be blurry; data entities can be located anywhere and be complex, while contextual entities can have a Web presence beyond their description inside the RO-Crate. For instance `https://orcid.org/0000-0002-1825-0097` is primarily an identifier for a person, but secondarily also a web page that describes that person and their academic work.

Any particular IRI might appear as a contextual entity in one RO-Crate and as a data entity in another; their distinction lies in the fact that data entities can be considered to be *contained* or captured by the RO-Crate, while contextual entities mainly *explain* the RO-Crate and its entities. It follows that an RO-Crate should have one or more data entities, although this is not a formal requirement.

Figure **??** shows a UML view of RO-Crate, highlighting the different types of data entities and contextual entities that can be aggregated and related.

fig:uml width="90

2.1.4. Best Practice Guide rather than strict specification

RO-Crate builds on Linked Data standards (JSON-LD [**?**]) and community-evolved vocabularies (Schema.org[6] [**?**]). RO-Crate aims to build a set of best practices on how to practically apply these

[4]<https://www.researchobject.org/ro-crate/1.1/root-data-entity.html#direct-properties-of-the-root-data-entity>
[5]<https://www.researchobject.org/ro-crate/1.1/contextual-entities.html#contextual-vs-data-entities>
[6]<https://schema.org/>

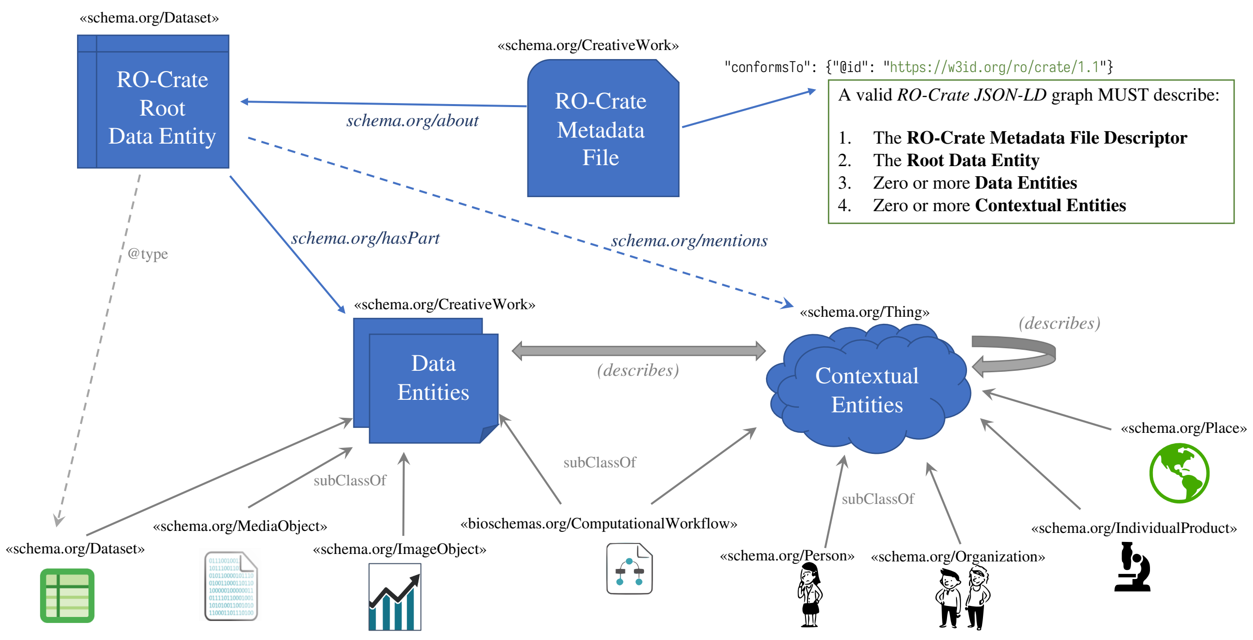
«schema.org/Dataset»
RO-Crate Root Data Entity
«schema.org/CreativeWork»
RO-Crate Metadata File
schema.org/about
"conformsTo": {"@id": "https://w3id.org/ro/crate/1.1"}
A valid RO-Crate JSON-LD graph MUST describe:
1. The RO-Crate Metadata File Descriptor
2. The Root Data Entity
3. Zero or more Data Entities
4. Zero or more Contextual Entities
@type
schema.org/hasPart
schema.org/mentions
«schema.org/CreativeWork»
Data Entities
(describes)
«schema.org/Thing»
Contextual Entities
(describes)
subClassOf
subClassOf
subClassOf
«schema.org/Place»
«schema.org/Dataset»
«schema.org/MediaObject»
«schema.org/ImageObject»
«bioschemas.org/ComputationalWorkflow»
«schema.org/Person»
«schema.org/Organization»
«schema.org/IndividualProduct»

Fig. 2. **UML model view of RO-Crate.** The *RO-Crate Metadata File* conforms to a version of the specification, which mainly describes the *RO-Crate Root Data Entity* representing the Research Object as a dataset. The RO-Crate aggregates *data entities* (`hasPart`) which are further described using *contextual entities*. Multiple types and relations from Schema.org allow annotations to be more specific, including figures, nested datasets, computational workflows, people, organisations, instruments and places.

existing standards in a common way to describe research outputs and their provenance, without having to learn each of the underlying technologies in detail.

As such, the RO-Crate 1.1[7] specification [**?**] can be seen as an opinionated and example-driven guide to writing Schema.org metadata as JSON-LD, which leaves it open for implementers to include additional metadata using other Schema.org types and properties, or even additional Linked Data vocabularies/ontologies or their own ad-hoc terms.

2.1.5. Checking Simplicity

As previously stated, one aim of RO-Crate is to be conceptually simple. This simplicity has been repeatedly checked and confirmed through a community review process. For instance, in the discussion on supporting ad-hoc vocabularies[8] in RO-Crate, the community explored potential Linked Data solutions. The conventional wisdom in RDF best practice[9] is to establish a vocabulary with a new URI namespace, formalised using RDF Schema[10] or OWL[11] ontologies, however, as this may seem excessive learning curve for the use case of clarifying that `sentence` in an RO-Crate refers to prisoner sentencing rather than a linguistic structure, the RO-Crate community instead agreed

[7]<https://w3id.org/ro/crate/1.1>
[8]<https://github.com/ResearchObject/ro-crate/issues/71>
[9]<https://www.w3.org/TR/swbp-vocab-pub/>
[10]<http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>
[11]<http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>

on a dual lightweight approach: (?) Document[12] how projects with their own web-presence can make a pure HTML-based vocabulary, and (?) provide a community-wide PID namespace under https://w3id.org/ro/terms/[13] that redirect to simple CSV files maintained in GitHub[14].

To further verify this idea, we have formalised the RO-Crate definition (see Appendix A). An important result of this exercise is that the underlying data structure of RO-Crate, although conceptually a graph, is represented as a tree limited to depth 2. The formalisation also emphasises the *boundedness* of the structure; namely, the fact that elements are specifically identified as being either semantically *contained* by the RO-Crate (`hasPart`) or mainly referenced (`mentions`) and typed as *external* to the Research Object (Contextual Entities).

2.2. Technical implementation of the model

The RO-Crate conceptual model has been realised using JSON-LD and Schema.org in a prescriptive form as discussed in the best practice approach. This technical approach again caters for simplicity.

JSON-LD[15] provides a way to express Linked Data as a JSON structure, where a *context* provides mapping to RDF properties and classes. While JSON-LD cannot map arbitrary JSON structures to RDF, we found it does lower the barrier regarding Linked Data serialisations as it follows a JSON structure, nowadays a very common and popular format for data exchange.

However, JSON-LD alone, has too many degrees of freedom and hidden complexities for software developers to reliably produce and consume without specialised expertise or software libraries. A large part of the RO-Crate specification is therefore dedicated to describing JSON structures.

2.2.1. RO-Crate JSON-LD

RO-Crate mandates the use of flattened, compacted JSON-LD[16] where a single `@graph` array contains all the data and contextual entities in a flat list. An example can be seen in the JSON-LD snippet below, describing a simple RO-Crate containing two datasets (data1.txt and data2.txt):

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    { "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "about": {"@id": "./"},
    },
    { "@id": "./",
      "@type": "Dataset",
      "hasPart": [
        {"@id": "data1.txt"},
        {"@id": "data2.txt"}
      ]
    },
```

[12]<https://www.researchobject.org/ro-crate/1.1/appendix/jsonld.html#adding-new-or-ad-hoc-vocabulary-terms>
[13]<https://w3id.org/ro/terms/>
[14]<https://github.com/ResearchObject/ro-terms>
[15]<https://json-ld.org/>
[16]<https://www.researchobject.org/ro-crate/1.1/appendix/jsonld.html>

```
    { "@id": "data1.txt",
      "@type": "File",
      "author": {"@id": "#alice"},
    },
    { "@id": "data2.txt",
      "@type": "File",
      "author": {"@id": "#alice"},
    },
    { "@id": "#alice",
      "@type": "Person",
      "name": "Alice",
    },
    { "@id": "http://sws.geonames.org/8152662/",
      "@type": "Place",
      "name": "Catalina Park"
    }
] }
```

Figure X: *Simplified RO-Crate JSON-LD showing the flattened compacted `@graph` array*

It can be argued that this is a more graph-like approach than the tree structure JSON would otherwise invite, and which is normally emphasised as a feature of JSON-LD in order to “hide” its RDF nature.

However, we found that the use of trees for, e.g. *a* Person *entity appearing as author of a `File` which nests under a `Dataset`, `hasPart`*, counter-intuitively leads one to consider the JSON-LD as an RDF Graph, since an identified `Person` entity can then appear at multiple and repeated points of the tree (e.g. author of multiple files), necessitating node merging or duplication.

By comparison, a single flat `@graph` array approach means that applications can process and edit each entity as pure JSON by a simple lookup based on `@id`. At the same time, lifting all entities to the same level emphasises Research Object's principle that describing the context and provenance is just as important as describing the data.

RO-Crate reuses Schema.org, but provides its own versioned JSON-LD context, which has a similar flat list with the mapping from JSON-LD keys to their URI equivalents (e.g. `author` maps to http://schema.org/author[17]). The rationale behind this decision is to support JSON-based RO-Crate applications that are largely unaware of JSON-LD, and thus still may want to process the `@context` to find Linked Data definitions of unknown properties and types. Not reusing the official Schema.org context means RO-Crate is also able to map in additional vocabularies where needed, namely the *Portland Common Data Model* (PCDM) [**?**] for repositories and Bioschemas [**?**] for describing computational workflows.

Similarly, rather than relying on implications from `"@type": "@id"` annotations in the context, RO-Crate JSON-LD distinguishes explicitly between references to other entities (`{"@id": "#alice"}`) and string values (`"Alice"`) - meaning RO-Crate applications can find the corresponding entity without parsing the `@context`.

[17]<http://schema.org/author>

2.3. RO-Crate Community

The RO-Crate conceptual model, implementation and best practices are developed by a growing community of researchers, developers and publishers. The RO-Crate community is a key aspect of its effectiveness in making research artefacts FAIR. Fundamentally, the community provides the overall context of the implementation and model and ensures its interoperability.

The RO-Crate community consists of:

1. a diverse set of people representing a variety of stakeholders;
2. a set of collective norms;
3. an open platform that facilitates communication (GitHub, Google Docs, monthly teleconferences).

2.3.1. People

The initial concept of RO-Crate was formed at the first Workshop on Research Objects (RO2018[18]), held as part of the IEEE conference on eScience. This workshop followed up on considerations made at a Research Data Alliance (RDA) meeting on Research Data Packaging[19] that found similar goals across multiple data packaging efforts [**?**]: simplicity, structured metadata and the use of JSON-LD.

An important outcome of discussions that took place at RO2018 was the conclusion that the original Wf4Ever Research Object ontologies [**?**], in principle sufficient for packaging research artefacts with rich descriptions, were, in practice, considered inaccessible for regular programmers (e.g. web developers) and in danger of being incomprehensible for domain scientists due to their reliance on Semantic Web technologies and other ontologies.

DataCrate [**?**] was presented at RO2018 as a promising lightweight alternative approach, and an agreement was made by a group of volunteers to attempt building "RO Lite" as a combination of DataCrate's implementation and Research Object's principles.

This group, originally made up of library and Semantic Web experts, has subsequently grown to include domain scientists, developers, publishers and more. This perspective of multiple views led to the specification being used in a variety of domains, from bioinformatics and regulatory submissions to humanities and cultural heritage preservation.

The RO-Crate community is strongly engaged with the European-wide biology/bioinformatics collaborative e-Infrastructure, ELIXIR, [**?**], along with European Open Science Cloud (EOSC) projects including EOSC-Life and FAIRplus. RO-Crate has also established collaborations with Bioschemas, GA4GH, OpenAIRE and multiple H2020 projects.

A key set of stakeholders are developers; the RO-Crate community has made a point of attracting developers who can implement the specifications but, importantly, keeps "developer user experience" in mind. This means that the specifications are straightforward to implement and thus do not require expertise in technologies that are not widely deployed.

This notion of catering to "developer user experience" is an example of the set of norms that have developed and now define the community.

2.3.2. Norms

The RO-Crate community is driven by conventions or notions that are prevalent within it but not formalised. Here, we distil what we as authors believe are the critical set of norms that have facilitated the development of RO-Crate and contributed to the ability for RO-Crate research packages to

[18]<https://www.researchobject.org/ro2018/>

[19]<https://rd-alliance.org/approaches-research-data-packaging-rda-11th-plenary-bof-meeting>

be FAIR. This is not to say that there are no other norms within the community or that everyone in the community holds these uniformly. Instead, what we emphasise is that these norms are helpful and also shaped by community practices.

1. Simplicity
2. Developer friendliness
3. Focus on examples and best practices rather than rigorous specification
4. Reuse “just enough” Web standards

A core norm of RO-Crate is that of **simplicity**, which sets the scene for how we guide developers to structure metadata with RO-Crate. We focus mainly on documenting simple approaches to the most common use cases, such as authors having an affiliation. This norm also influences our take on **developer friendliness**; for instance, we are using the Web-native JSON format, allowing only a few of JSON-LD’s flexible Linked Data features. Moreover, the RO-Crate documentation is largely built up by **examples** showcasing **best practices**, rather than rigorous specifications. In a sense, we are allowed to do this by building on existing **Web standards** that themselves are defined rigorously, which we utilise ***“just enough”*** in order to benefit from the advantages of Linked Data (e.g. extensions by namespaced vocabularies), without imposing too many developer choices or uncertainties (e.g. having to choose between the many RDF syntaxes).

While the above norms alone could easily lead to the creation of “yet another” JSON format, we keep the goal of **FAIR interoperability** of the captured metadata, and therefore follow closely FAIR best practices and current developments such as data citations, PIDs, open repositories and recommendations for sharing research outputs and software.

2.3.3. Open Platforms

The critical infrastructure that enables the community around RO-Crate is the use of open development platforms. This underpins the importance of open community access to supporting FAIR. Specifically, it is difficult to build and consume FAIR research artefacts without being able to access the specifications, understand how they are developed, know about any potential implementation issues, and discuss usage to evolve best practices.

It was seen as more important to document real-life examples and best practices rather than developing a rigorous specification. At the same time, we agreed to be opinionated on the syntactic form to reduce the jungle of implementation choices; we wanted to keep the important aspects of Linked Data to adhere to the FAIR principles while retaining the option of combining and extending the structured metadata using the existing Semantic Web stack, not just build “yet another” standalone JSON format.

Further work during 2019 started adapting the DataCrate documentation through a more collaborative and exploratory *RO-Lite* phase, initially using Google Docs for review and discussion, then moving to GitHub as a collaboration space for developing what is now the RO-Crate specification, maintained as Markdown[20] in GitHub Pages and published through Zenodo.

In addition to the typical Open Source-style development with GitHub issues and pull requests, the RO-Crate Community now has two regular monthly calls, a Slack channel and a mailing list for coordinating the project, and many of its participants collaborate on RO-Crate at multiple conferences and coding events such as the ELIXIR BioHackathon. The community is jointly developing the RO-Crate specification and Open Source tools, as well as providing support and considering

[20]<https://github.com/researchobject/ro-crate/>

new use cases. The RO-Crate Community[21] is open for anyone to join, to equally participate under a code of conduct, and currently has more than 40 members.

Tool name *Brief Description*	Targets	Language / Platform	Status
Describo [?]	Research Data Managers	NodeJS (Desktop)	RC
Interactive desktop application to create, update and export RO-Crates for different profiles			
Describo Online [?]	Platform developers	NodeJS (Web)	Alpha
Web-based application to create RO-Crates using cloud storage			
ro-crate-excel [?]	Data managers	JavaScript	
Command-line tool to help create RO-Crates and HTML-readable rendering			
ro-crate-html-js [?]	Developers	JavaScript	Beta
HTML rendering of RO-Crate			
ro-crate-js [?]	Research Data Managers	JavaScript	Alpha
Library for creating/manipulating crates; basic validation code			
ro-crate-ruby [?]	Developers	Ruby	Beta
Ruby library for reading/writing RO-Crate, with workflow support			
ro-crate-py [?]	Developers	Python	Beta
Object-oriented Python library for reading/writing RO-Crate			
WorkflowHub [?]	Workflow users	Ruby	Beta
Workflow repository; imports and exports Workflow RO-Crate			
Life Monitor [?]	Workflow developers	Python	Alpha
Workflow testing and monitoring service; Workflow Testing profile of RO-Crate			
SCHeMa [?]	Workflow users	PHP	Alpha
Workflow execution using RO-Crate as exchange mechanism			
galaxy2cwl [?]	Workflow developers	Python	Alpha
Wraps Galaxy workflow as Workflow RO-Crate			
Modern PARADISEC [?]	Repository managers	Platform	Beta
Cultural Heritage portal based on OCFL and RO-Crate			
ONI express [?]	Repository managers	Platform	Beta
Platform for publishing data and documents stored in an OCFL repository via a web interface			
ocfl-tools [?]	Developers	JavaScript (CLI)	Beta
Tools for managing RO-Crates in an OCFL repository			
RO Composer [?]	Repository developers	Java	Alpha
REST API for gradually building ROs for given profile			
RDA maDMP Mapper [?]	Data Management Plan users	Python	Beta
Mapping between machine-actionable data management plans (maDMP) and RO-Crate[?]			
Ro-Crate_2_ma-DMP [?]	Data Management Plan users	Python	Beta
Convert between machine-actionable data management plans (maDMP) and RO-Crate			
CheckMyCrate [?]	Developers	Python (CLI)	Alpha
Validation according to Workflow RO-Crate profile			

Table 1

Applications and libraries implementing RO-Crate, targeting different types of users across multiple programming languages. Status is indicative as assessed by this work (Alpha < Beta < Release Candidate (RC) < Release).

[21]<https://www.researchobject.org/ro-crate/community>

3. RO-Crate Tooling

The work of the community led to the development of a number of tools for creating and using RO-Crates. Table **??** shows the current set of implementations. Reviewing this list, one can see that tools support commonly used programming languages, including Python, JavaScript, and Ruby. Additionally, these tools can be integrated into commonly used research environments; in particular, the command line (*ro-crate-html-js*). Furthermore, there are tools that cater to the end-user (*Describo*, *Workflow Hub*). For example, Describo was developed to help researchers of the Australian Criminal Characters project[22] annotate historical prisoner records to gain greater insight into the history of Australia [**?**].

While the development of these tools is promising, our analysis of their maturity status shows that the majority of them are in the Beta stage. This is partly due to the fact that the RO-Crate specification itself only recently reached 1.0 status, in November 2019 [**?**]. Now that there is a fixed point of reference, and RO-Crate 1.1 (October 2020) [**?**] has stabilised based on feedback from application development, we expect to see a further increase in the maturity of these tools, along with the creation of new ones.

Given the stage of the specification, these tools have been primarily targeted to developers, essentially providing them with the core libraries for working with RO-Crates. Another target has been that of research data managers who need to manage and curate large amounts of data.

4. Profiles of RO-Crate in use

RO-Crate is fundamentally an infrastructure to help build FAIR research artefacts. In other words, the key question is whether RO-Crate can be used to share and (re)use research artefacts. Here we look at three research domains where RO-Crate is being applied: Bioinformatics, Regulatory Science and Cultural Heritages. In addition, we note how RO-Crate may have an important role as part of machine-actionable data management plans and institutional repositories.

From these varied uses of RO-Crate we observe a natural differences in their detail level and the type of entities described by the RO-Crate. For instance, on submission of an RO-Crate to a workflow repository, it is reasonable to expect the RO-Crate to contain at least one workflow, ideally with a declared licence and workflow language. Specific additional recommendations such as on identifiers is also needed to meet the emerging requirements of FAIR Digital Objects[23]. Work has now begun[24] to formalise these different *profiles* of RO-Crates, which may impose additional constraints based on the needs of a specific domain or use case.

4.1. Bioinformatics workflows

WorkflowHub.eu[25] is a European cross-domain registry of computational workflows, supported by European Open Science Cloud projects, e.g. EOSC-Life[26], and research infrastructures including the pan-European bioinformatics network ELIXIR[27] [**?**]. As part of promoting workflows as

[22]<https://criminalcharacters.com/>
[23]<https://fairdo.org/>
[24]<https://github.com/ResearchObject/ro-crate/issues/153>
[25]<https://workflowhub.eu/>
[26]<https://www.eosc-life.eu/>
[27]<https://elixir-europe.org/>

reusable tools, WorkflowHub includes documentation and high-level rendering of the workflow structure independent of its native workflow definition format. The rationale is that a domain scientist can browse all relevant workflows for their domain, before narrowing down their workflow engine requirements. As such, the WorkflowHub is intended largely as a registry of workflows already deposited in repositories specific to particular workflow languages and domains, such as UseGalaxy.eu [**?**] and Nextflow nf-core [**?**].

We here describe three different RO-Crate profiles developed for use with WorkflowHub.

4.1.1. Profile for describing workflows

Being cross-domain, WorkflowHub has to cater for many different workflow systems. Many of these, for instance Nextflow [**?**] and Snakemake [**?**], by virtue of their script-like nature, reference multiple neighbouring files typically maintained in a GitHub repository. This calls for a data exchange method that allows keeping related files together. WorkflowHub has tackled this problem by adopting RO-Crate as the packaging mechanism [**?**], typing and annotating the constituent files of a workflow and — crucially — marking up the workflow language, as many workflow engines use common file extensions like `*.xml` and `*.json`. Workflows are further described with authors, licence, diagram previews and a listing of their inputs and outputs. RO-Crates can thus be used for interoperable deposition of workflows to WorkflowHub, but are also used as an archive for downloading workflows, embedding metadata registered with the WorkflowHub entry and translated workflow files such as abstract Common Workflow Language (CWL) [**?**] definitions and diagrams [**?**].

RO-Crate acts therefore as an interoperability layer between registries, repositories and users in WorkflowHub. The iterative development between WorkflowHub developers and the RO-Crate community heavily informed the creation of the Bioschemas [**?**] profile for Computational Workflows[28], which again informed the RO-Crate 1.1 specification on workflows[29] and led to the RO-Crate Python library [**?**] and WorkflowHub's **Workflow RO-Crate profile**[30], which, in a similar fashion to RO-Crate itself, recommends which workflow resources and descriptions are required. This co-development across project boundaries exemplifies the drive for simplicity and for establishing best practices.

4.1.2. Profile for recording workflow runs

While RO-Crates in WorkflowHub so far have been focused on workflows that are ready to be run, development of WorkflowHub are now creating a **Workflow Run RO-Crate profile** for the purposes of benchmarking, testing and executing workflows. As such, RO-Crate serves as a container of both a *workflow definition* that may be executed and of a particular *workflow execution with test results*. This profile is a continuation of our previous work with capturing workflow provenance in a Research Object in CWLProv [**?**] and TavernaPROV [**?**]. In both cases, we used the PROV Ontology [**?**], including details of every task execution with all the intermediate data, which required significant workflow engine integration[31]. To simplify from that approach, for this Workflow Run RO-Crate profile we will use a higher level schema.org provenance[32] for the input/output boundary

[28]<https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE/>

[29]<https://www.researchobject.org/ro-crate/1.1/workflows.html>

[30]<https://about.workflowhub.eu/Workflow-RO-Crate/>

[31]CWLProv and TavernaProv predate RO-Crate, but use RO-Bundle[**?**], a similar Research Object packaging method with JSON-LD metadata.

[32]<https://www.researchobject.org/ro-crate/1.1/provenance.html#software-used-to-create-files>

of the overall workflow execution. This *Level 1 workflow provenance* [**?**] can be expressed generally across workflow languages with minimal engine changes, with the option of more detailed provenance traces as separate PROV resources in the RO-Crate.

WorkflowHub has recently enabled minting of Digital Object Identifiers (DOIs), a PID commonly used for scholarly artefacts, for registered workflows, e.g. `10.48546/workflowhub.workflow.56.1` [**?**], lowering the barrier for citing workflows as computational methods along with their FAIR metadata – captured within an RO-Crate. While it is not an aim for WorkflowHub to be a repository of workflow runs and their data, RO-Crates of *exemplar workflow runs* serve as useful workflow documentation, as well as being an exchange mechanism that preserve FAIR metadata in a diverse workflow execution environment.

4.1.3. Profile for testing workflows

The value of computational workflows, however, is potentially undermined by the ”collapse” over time of the software and services they depend upon: for instance, software dependencies can change in a non-backwards-compatible manner, or active maintenance may cease; an external resource, such as a reference index or a database query service, could shift to a different URL or modify its access protocol; or the workflow itself may develop hard-to-find bugs as it is updated. This can take a big toll on the workflow's reusability and on the reproducibility of any processes it evokes.

For this reason, WorkflowHub is complemented by a monitoring and testing service called LifeMonitor[**?**], also supported by EOSC-Life. LifeMonitor's main goal is to assist in the creation, periodic execution and monitoring of workflow tests, enabling the early detection of software collapse in order to minimise its detrimental effects. The communication of metadata related to workflow testing is achieved through the adoption of a **Workflow Testing RO-Crate profile**[33] stacked on top of the *Workflow RO-Crate* profile. This further specialisation of Workflow RO-Crate allows to specify additional testing-related entities (test suites, instances, services, etc.), leveraging RO-Crate's extension mechanism[34] through the addition of terms from custom namespaces.

In addition to showcasing RO-Crate's extensibility, the testing profile is an example of the format's flexibility and adaptability to the different needs of the research community. Though ultimately related to a computational workflow, in fact, most of the testing-specific entities are more about describing a protocol for interacting with a monitoring service than a set of research outputs and its associated metadata. Indeed, one of LifeMonitor's main functionalities is monitoring and reporting on test suites running on existing Continuous Integration (CI) services, which is described in terms of service URLs and job identifiers in the testing profile. In principle, in this context, data could disappear altogether, leading to an RO-Crate consisting entirely of contextual entities. Such an RO-Crate acts more as an exchange format for communication between services (WorkflowHub and LifeMonitor) than as an aggregator for research data and metadata, providing a good example of the format's high versatility.

4.2. Regulatory Sciences

BioCompute Objects[35] (BCO) [**?**] is a community-led effort to standardise submissions of computational workflows to biomedical regulators. For instance, a genomics sequencing pipeline, as part

[33]<https://crs4.github.io/life_monitor/workflow_testing_ro_crate>
[34]<https://www.researchobject.org/ro-crate/1.1/appendix/jsonld.html#extending-ro-crate>
[35]<https://www.biocomputeobject.org/>

of a personalised cancer treatment study, can be submitted to the US Food and Drugs Administration (FDA) for approval. BCOs are formalised in the standard IEEE 2791-2020 [**?**] as a combination of JSON Schemas[36] that define the structure of JSON metadata files describing exemplar workflow runs in detail, covering aspects such as the usability and error domain of the workflow, its runtime requirements, the reference datasets used and representative output data produced.

BCOs provide a structured view over a particular workflow, informing regulators about its workings independently of the underlying workflow definition language. However, BCOs have only limited support for additional metadata[37]. For instance, while the BCO itself can indicate authors and contributors, and in particular regulators and their review decisions, it cannot describe the provenance of individual data files or workflow definitions.

As a custom JSON format, BCOs cannot be extended with Linked Data concepts, except by adding an additional top-level JSON object formalised in another JSON Schema. A BCO and workflow submitted by upload to a regulator will also frequently consist of multiple cross-related files. Crucially, there is no way to tell whether a given `*.json` file is a BCO file, except by reading its content and check for its `spec_version`.

We can then consider how a BCO and its referenced artefacts can be packaged and transferred following FAIR principles. **BCO RO-Crate**[38][**?**], part of the BioCompute Object user guides, defines a set of best practices for wrapping a BCO with a workflow, together with its exemplar outputs in an RO-Crate, which then provides typing and additional provenance metadata of the individual files, workflow definition, referenced data and the BCO metadata itself.

Here the BCO is responsible for describing the *purpose* of a workflow and its run at an abstraction level suitable for a domain scientist, while the more open-ended RO-Crate describes the surroundings of the workflow, classifying and relating its resources and providing provenance of their existence beyond the BCO. This emerging *separation of concerns* highlight how RO-Crate is used side-by-side of existing standards, even where there are apparent partial overlaps.

A similar separation of concerns can be found if considering the RO-Crate as a set of files, where the *transport-level* metadata, such as checksum of files, are delegated to BagIt[39] manifests, a standard focusing on the preservation challenges of digital libraries[**?**]. As such, RO-Crates are not required to iterate all the files in their folder hierarchy, only those that benefit from being described.

Specifically, a BCO alone is insufficient for reliable re-execution of a workflow, which would need a compatible workflow engine depending on the workflow definition language, so IEEE 2791 recommends using Common Workflow Language [**?**] for interoperable pipeline execution. CWL itself relies on tool packaging in software containers using Docker[40] or Conda[41]. Thus, we can consider BCO RO-Crate as a stack: transport-level manifests of files (BagIt), provenance, typing and context of those files (RO-Crate), workflow overview and purpose (BCO), interoperable workflow definition (CWL) and tool distribution (Docker).

[36]<https://opensource.ieee.org/2791-object/ieee-2791-schema/>
[37]IEEE 2791-2020 do permit user extensions in the *extension domain* by referencing additional JSON Schemas.
[38]<https://biocompute-objects.github.io/bco-ro-crate/>
[39]<https://www.researchobject.org/ro-crate/1.1/appendix/implementation-notes.html#adding-ro-crate-to-bagit>
[40]<https://www.docker.com/>
[41]<https://docs.conda.io/>

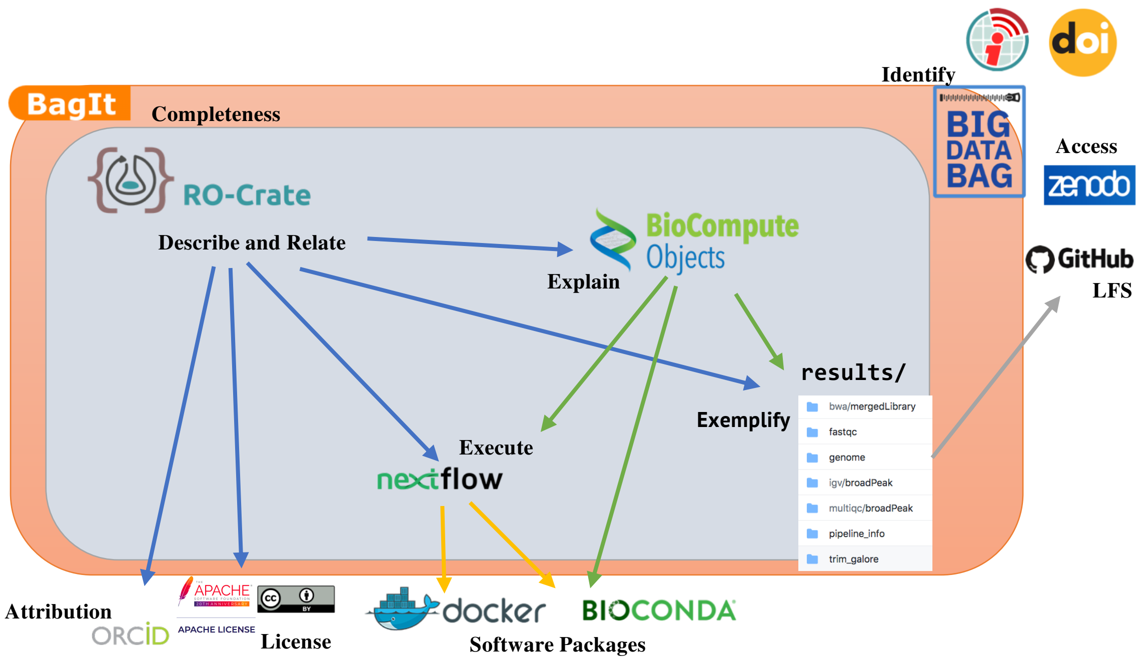

Fig. 3. **Separation of Concerns in BCO RO-Crate**. BioCompute Object (IEEE2791) is a JSON file that structurally explains the purpose and implementation of a computational workflow, for instance implemented in Nextflow, that installs the workflow's software dependencies as Docker containers or BioConda packages. An example execution of the workflow shows the different kinds of result outputs, which may be external, using GitHub LFS to support larger data. RO-Crate gathers all these local and external resources, relating them and giving individual descriptions, for instance permanent DOI identifiers for reused datasets accessed from Zenodo, but also adding external identifiers to attribute authors using ORCID or to identify which licences apply to individual resources. The RO-Crate and its local files are captured in a BagIt whose checksum ensures completeness, combined with Big Data Bag [**?**] features to "complete" the bag with large external files such as the workflow outputs

4.3. Digital Humanities: Cultural Heritage

PARADISEC[42] (the Pacific And Regional Archive for Digital Sources in Endangered Cultures) maintains a repository of more than 500,000 files documenting endangered languages across more than 16,000 items, collected over many years by researchers interviewing and recording native speakers across the region. As a proposed update of the 18 year old infrastructure, the Modern PARADISEC demonstrator[43] has been developed[44] to also help digitally preserve these artefacts using the Oxford Common File Layout[45] (OCFL) for file consistency and RO-Crate for structuring and capturing the metadata of each item. The existing PARADISEC data collection has been ported and captured as RO-Crates. A web portal then exposes the repository and its entries by indexing the RO-Crate metadata files using Elasticsearch as a "NoSQL" object database, presenting a domain-specific view of the items - the RO-Crate is "hidden" and does not change the user interface.

[42]<https://www.paradisec.org.au/>
[43]<https://mod.paradisec.org.au/>
[44]<https://arkisto-platform.github.io/case-studies/paradisec/>
[45]<https://ocfl.io/1.0/spec/>

This use case takes advantage of several RO-Crate features and principles. Firstly, the transcribed metadata are now independent of the PARADISEC platform and can be archived, preserved and processed in its own right, using Schema.org vocabularies augmented with PARADISEC-specific terms. The lightweight infrastructure with RO-Crate as the holder of itemised metadata in regular files (organised using OCFL[**?**], with checksums for integrity checking and versioning) also gives flexibility for future developments and maintenance; for example, potentially using Linked Data software such as a graph database, queried using SPARQL triple patterns across RO-Crates, or a “last resort” fallback to the generic RO-Crate HTML preview [**?**], which can be hosted as static files by any web server, in line with the approach taken by the Endings Project[46].

4.4. Machine-actionable Data Management Plans

Machine-actionable Data Management Plans (maDMPs) have been proposed as an improvement to automate FAIR data management tasks in research [**?**], e.g. by using PIDs and controlled vocabularies to describe what happens to data over the research life cycle [**?**]. The Research Data Alliance’s *DMP Common Standard* for maDMPs [**?**] is one such formalisation for expressing maDMPs, which can be expressed as Linked Data using the DMP Common Standard Ontology [**?**], a specialisation of the W3C Data Catalog Vocabulary (DCAT) [**?**]. RDA maDMPs are usually expressed using regular JSON, conforming to the DMP JSON Schema.

A mapping has been produced between Research Object Crates and Machine-actionable Data Management Plans [**?**], implemented by the RO-Crate RDA maDMP Mapper [**?**]. A similar mapping has been implemented by `RO-Crate_2_ma-DMP` [**?**]. In both cases, a maDMP can be converted to a RO-Crate, or vice versa. In [**?**] this functionality caters for two use cases:

1. Start a skeleton data management plan based on an existing RO-Crate dataset, e.g. from an RO-Crate from WorkflowHub.
2. Instantiate an RO-Crate based on a data management plan.

An important difference here is that data management plans are (ideally) written in advance of data production, while RO-Crates are typically created to describe data after it has been generated. This approach shows the importance of *templating* to make both tasks more automatable and achievable, and how RO-Crate can fit into earlier stages of the research life cycle.

4.5. Institutional data repositories – Harvard Data Commons

The concept of a Data Commons for research collaboration was originally defined as ”cyber-infrastructure that co-locates data, storage, and computing infrastructure with commonly used tools for analysing and sharing data to create an interoperable resource for the research community” [**?**]. More recently, it was established to integrate active data-intensive research with data management and archival best practices. It facilitates research by providing computational infrastructure where researchers can use, share and store data, software, workflows and other digital artefacts used in their studies. Furthermore, the Commons feature tools and services, such as computation clusters and storage for scalability, data repositories for disseminating and preserving regular, but also large or sensitive datasets, and other research assets. Multiple initiatives were undertaken to create

[46]The Endings Project https://endings.uvic.ca/ is a five-year project funded by the Social Sciences and Humanities Research Council (SSHRC) that is creating tools, principles, policies and recommendations for digital scholarship practitioners to create accessible, stable, long-lasting resources in the humanities.

Data Commons on national, research, and institutional levels. For example, the Australian Research Data Commons (ARDC)[47] [**?**] is a national initiative that enables local researchers and industries to access computing infrastructure, training, and curated datasets for data-intensive research. NCI's Genomic Data Commons[48] (GDC) [**?**] provides the cancer research community with access to a vast volume of genomic and clinical data. Initiatives such as Research Data Alliance (RDA) Global Open Research Commons[49] propose standards on the implementation of Data Commons to avoid them becoming ”data silos” and enable interoperability from one Data Commons to another.

Harvard Data Commons [**?**] aims to address data access and reuse challenges of cross-disciplinary research within a research institution. It brings together multiple institutional schools, libraries, computing centres and the Harvard Dataverse data repository[50]. Dataverse[51] [**?**] is a free and open-source software platform to archive, share and cite research data. The Harvard Dataverse repository is the largest of 70 installations worldwide, containing over 100K datasets with about 1M data files. Toward the goal of facilitating collaboration and data discoverability and management within the university, Harvard Data Commons has the following primary objectives:

1. integrating Harvard Research Computing with Harvard Dataverse by leveraging Globus endpoints [**?**] that will allow an automatic transfer of large datasets to the repository. In some cases, only the metadata will be transferred while the data stays stored in remote storage;
2. supporting advanced research workflows and providing packaging options for assets such as code and workflows in the Harvard Dataverse repository to enable reproducibility and reuse, and
3. integrating repositories supported by Harvard, which are DASH[52], the open access institutional repository, the Digital Repository Services (DRS) for preserving digital asset collections, and the Harvard Dataverse.

Particularly relevant to this paper is the second objective of the Harvard Data Commons, which aims to support the deposit of research artefacts to Harvard Dataverse with sufficient information in the metadata to allow their future reuse (Figure **??**). Considering the requirements of incorporating data, code, and other artefacts from various institutional infrastructures, Harvard Data Commons is currently working on RO-Crate adaptation. The RO-Crate metadata provides the necessary structure to make all research artefacts FAIR. The Dataverse software already has extensive support for metadata, including the Data Documentation Initiative (DDI), Dublin Core, DataCite, and Schema.org. Incorporating RO-Crate, which has the flexibility to describe a wide range of research resources, will facilitate their seamless transition from one infrastructure to the other within the Harvard Data Commons.

fig:hdc

Even though the Harvard Data Commons is specific to Harvard University, the overall vision and the three objectives can be abstracted and applied to other universities or research organisations. The Commons will be designed and implemented using standards and commonly-used approaches to make it interoperable and reusable by others.

[47]<https://ardc.edu.au>
[48]<https://gdc.cancer.gov/>
[49]<https://www.rd-alliance.org/groups/global-open-research-commons-ig>
[50]<https://dataverse.harvard.edu/>
[51]<https://dataverse.org/>
[52]<https://dash.harvard.edu>

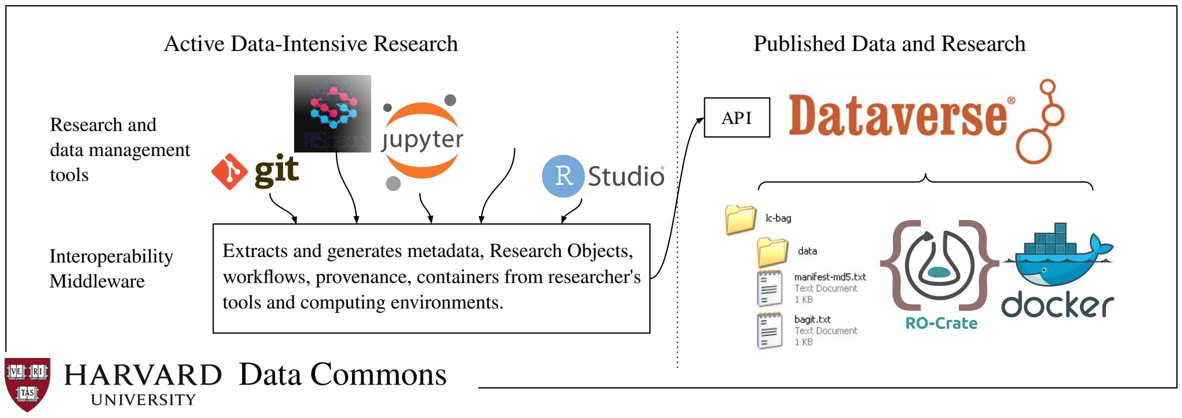
Active Data-Intensive Research
Published Data and Research
Research and data management tools
git
jupyter
RStudio
Interoperability Middleware
Extracts and generates metadata, Research Objects, workflows, provenance, containers from researcher's tools and computing environments.
API
Dataverse
lc-bag
data
manifest-md5.txt
bagit.txt
RO-Crate
docker
HARVARD UNIVERSITY Data Commons

Fig. 4. **One aspect of Harvard Data Commons**. Automatic encapsulation and deposit of artefacts from data management tools used during active research at the Harvard Dataverse repository.

5. Related Work

With the increasing digitalisation of research processes, there has been a significant call for the wider adoption of interoperable sharing of data and its associated metadata. For a comprehensive overview and recommendations, in particular for data, we refer to [**?**], which highlights the wide variety of metadata and documentation that the literature prescribes for enabling data reuse.

Here we focus on approaches for bundling research artefacts along with their metadata. This notion has a long history behind it [**?**], but recent approaches have followed three main strands: 1) publishing to centralised repositories; 2) packaging approaches similar to RO-Crate; and 3) bundling the computational workflow around a scientific experiment.

5.1. Bundling and Packaging Digital Research Artefacts

The challenge of describing computational workflows was one of the main motivations for the early proposal of *Research Objects* (RO) [**?**] as first-class citizens for sharing and publishing. The RO approach involves bundling datasets, workflows, scripts and results along with traditional dissemination materials like journal articles and presentations, forming a single package. Crucially, these resources are not just gathered, but also individually typed, described and related to each other using semantic vocabularies. As pointed out in [**?**] an open-ended *Linked Data* approach is not sufficient for scholarly communication: a common data model is also needed in addition to common and best practices for managing and annotating lifecycle, ownership, versioning and attributions.

Considering the FAIR principles, we can say with hindsight that the initial RO approaches strongly targeted *Interoperability*, with a particular focus on the reproducibility of *in-silico experiments* involving computational workflows and the reuse of existing RDF vocabularies.

The first implementation of Research Objects for sharing workflows in myExperiment [**?**] was based on RDF ontologies [**?**], building on Dublin Core, FOAF, SIOC, Creative Commons and OAI-ORE to form myExperiment ontologies for describing social networking, attribution and credit, annotations, aggregation packs, experiments, view statistics, contributions, and workflow components [**?**].

5.2. FAIR Digital Objects

FAIR Digital Objects (FDO) [**?**] have been proposed as a conceptual framework for making digital resources available in a Digital Objects (DO) architecture that encourages active use of the objects and their metadata. In particular, an FDO has five parts: (i) The FDO *content*, bit sequences stored in an accessible repository; (ii) a *Persistent Identifier* (PID) such as a DOI that identifies the FDO and can resolve these parts; (iii) Associated rich *metadata*, as separate FDOs; (iv) Type definitions, also separate FDOs; (v) Associated *operations* for the given types. A Digital Object typed as a Collection aggregates other DOs by reference.

As an ”abstract protocol[53]”, DOs could be implemented in multiple ways. One suggested implementation is the FAIR Digital Object Framework[54], based on HTTP and the Linked Data Principles. While there is agreement on using PIDs based on DOIs, consensus on how to represent common metadata, core types and collections as FDOs has not yet been reached. We argue that RO-Crate can play an important role for FDOs:

1. By providing a predictable and extensible serialisation of structured metadata.
2. By formalising how to aggregate digital objects as collections (and adding their context).
3. By providing a natural Metadata FDO in the form of the RO-Crate Metadata File.
4. By being based on Linked Data and schema.org vocabulary, meaning that PIDs already exist for common types and properties.

At the same time, it is clear that the goal of FDO is broader than that of RO-Crate; namely, FDOs are active objects with distributed operations, and add further constraints such as PIDs for every element. These features improve FAIR features of digital objects and are also useful for RO-Crate, but they also severely restrict the infrastructure that needs to be implemented and maintained in order for FDOs to remain available. RO-Crate, on the other hand, is more flexible: it can minimally be used within any file system structure, or ideally exposed through a range of Web-based scenarios. A *FAIR profile of RO-Crate* (e.g. enforcing PID usage) will fit well within a FAIR Digital Object ecosystem.

5.3. Packaging Workflows

The use of computational workflows, typically combining a chain of open source tools in an analytical pipeline, has gained prominence, in particular in the life sciences. Workflows may have initially been used to improve computational scalability, but they also assist in making computed data results FAIR [**?**], for instance by improving reproducibility [**?**], but also because programmatic data usage help propagate their metadata and provenance [**?**]. At the same time, however, workflows raise additional FAIR challenges, since they can be considered important research artefacts themselves, posing the problem of capturing and explaining the computational methods behind the analysis they perform [**?**].

Even when researchers follow current best practices for workflow reproducibility, [**?**] [**?**] the communication of outcomes through traditional academic publishing routes relying on a textual representation adds barriers that hinder reproducibility and FAIR use of the knowledge previously captured in the workflow.

[53]<https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf>
[54]<https://fairdigitalobjectframework.org/>

As a real-life example, let us look at a metagenomics article [**?**] where the authors have gone to extraordinary efforts to document the individual tools that have been reused, including their citations, versions, settings, parameters and combinations. The *Methods* section is 2 pages in tight double-columns with 24 additional references, supported by the availability of data on an FTP server (60 GB) [**?**] and of open source code in GitHub Finn-Lab/MGS-gut[55] [**?**], including the pipeline as shell scripts and associated analysis scripts in R and Python.

This attention to reporting detail for computational workflows is unfortunately not yet the norm, and although bioinformatics journals have strong *data availability* requirements, they frequently do not require authors to include or cite *software, scripts and pipelines* used for analysing and producing results [**?**] – rather, authors might be penalised for doing so [cite?] as it would detrimentally count against arbitrary limits on number of pages and references.

However detailed this additional information might be, another researcher who wants to reuse a particular computational method may first want to assess if the described tool or workflow is Re-runnable (executable at all), Repeatable (same results for original inputs on same platform), Reproducible (same results for original inputs with different platform or newer tools) and ultimately Reusable (similar results for different input data), Repurposable (reusing parts of the method for making a new method) or Replicable (rewriting the workflow following the method description). [**?**][**?**]

Following the textual description alone, researchers would be forced to jump straight to evaluate ”Replicable” by rewriting the pipeline from scratch. This can be expensive and error-prone. They would firstly need to install all the software dependencies and download reference datasets. This can be a daunting task in and of itself, which may have to be repeated multiple times as workflows typically are developed at small scale on desktop computers, scaled up to local clusters, and potentially put into production using cloud instances, each of which will have different requirements for software installations.

In recent years the situation has been greatly improved by software packaging and container technologies like Docker and Conda, which have seen increased adoption in life sciences [**?**] thanks to collaborative efforts such as BioConda [**?**] and BioContainers [**?**], and support by Linux distributions (e.g. Debian Med [**?**]). As of May 2021, more than 7000 software packages are available in BioConda alone[56], and 9000 containers in BioContainers[57]. Docker and Conda have gained integration in workflow systems such as Snakemake [**?**], Galaxy [**?**] and Nextflow [**?**], meaning a downloaded workflow definition can now be executed on a ”blank” machine (except for the workflow engine) with the underlying analytical tools installed on demand – but even here there is a reproducibility challenge, for instance Docker Hub's retention policy will expire container images after 6 months[58], or lack of recording versions of transitive dependencies of Conda packages could cause incompatibilities if the packages are subsequently updated. Except for brief metadata in their repositories, these containers and packages do not capture any semantic relationships of their content – rather their opaqueness and wrapping of arbitrary binary tools makes such relationships harder to find.

From this we see that computational workflows are themselves complex digital objects that needs to be recorded not just as files, but in the context of their execution environment, dependencies and

[55]<https://github.com/Finn-Lab/MGS-gut>
[56]<https://anaconda.org/bioconda/>
[57]<https://biocontainers.pro/#/registry>
[58]<https://www.docker.com/blog/docker-hub-image-retention-policy-delayed-and-subscription-updates/>

analytical purpose in research – as well as their FAIR metadata (e.g. version, license, attribution and identifiers).

6. Conclusion

RO-Crate provides a lightweight approach to packaging digital research artefacts with structured metadata, assisting developers and researchers to produce and consume FAIR data archives of their ROs, including PIDs, context and provenance of research artefacts. Aggregated data may be large and distributed, or located in regular folders on a file system.

As a set of best practice recommendations, developed by an open and broad community, RO-Crate shows how to use ”just enough” Linked Data standards in a consistent way, with structured metadata using a rich base vocabulary that can cover general-purpose contextual relations, whilst retaining extensibility to domain- and application-specific uses.

The adoption of simple web technologies in the RO-Crate specification has helped a rapid development of a wide variety of supporting open source tools and libraries. RO-Crate fits into the larger landscape of open scholarly communication and FAIR Digital Object infrastructure, and can be integrated into data repository platforms. RO-Crate can be applied as a data/metadata exchange mechanism, assist in long-term archival preservation of metadata and data, or simply used at small-scale by individual researchers. Thanks to its strong community support, new and improved profiles and tools are continuously added to the RO-Crate tooling landscape, making it easier for adopters to find examples and support for their own use case.

7. Acknowledgements

This work has received funding from the European Commission’s Horizon 2020 research and innovation programme for projects BioExcel-2[59] (H2020-INFRAEDI-2018-1 823830), IBISBA[60] (H2020-INFRAIA-2017-1-two-stage 730976, H2020-INFRADEV-2019-2 871118), EOSC-Life[61] (H2020-INFRAEOSC-2018-2 824087), SyntheSys+[62] (H2020-INFRAIA-2018-1 823827)

A. Trisovic is funded by the Alfred P. Sloan Foundation (grant number P-2020-13988). Harvard Data Commons is supported by an award from Harvard University Information Technology (HUIT).

7.1. Contributions

Author contributions to this article and the RO-Crate projet according to the Contributor Roles Taxonomy CASRAI CrEDiT[63] [**?**]:

Stian Soiland-Reyes Conceptualization, Data curation, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Software, Visualization, Writing – original draft, Writing – review & editing

Peter Sefton Conceptualization, Investigation, Methodology, Project administration, Resources, Software, Writing – review & editing

[59]<https://cordis.europa.eu/project/id/823830>
[60]<https://cordis.europa.eu/project/id/730976>
[61]<https://cordis.europa.eu/project/id/824087>
[62]<https://cordis.europa.eu/project/id/823827>
[63]<https://casrai.org/credit/>

Mercè Crosas Writing – review & editing
Leyla Jael Castro Methodology, Writing – review & editing
Frederik Coppens Writing – review & editing
José M. Fernández Methodology, Software, Writing – review & editing
Daniel Garijo Methodology, Writing – review & editing
Marco La Rosa Software, Methodology, Writing – review & editing
Simone Leo Software, Methodology, Writing – review & editing
Eoghan Ó Carragáin Investigation, Methodology, Project administration, Writing – review & editing
Marc Portier Methodology, Writing – review & editing
Ana Trisovic Software, Writing – review & editing
RO-Crate Community Investigation, Software, Validation, Writing – review & editing
Paul Groth Methodology, Supervision, Writing – original draft, Writing – review & editing
Carole Goble Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Visualization, Writing – review & editing

We would also like to thank for contributions from:

Finn Bacall Software, Methodology
Herbert Van de Sompel Writing – review & editing
Ignacio Eguinoa Software, Methodology
Nick Juty Writing – review & editing
Oscar Corcho Writing – review & editing
Stuart Owen Writing – review & editing
Laura Rodríguez-Navas Software, Visualization, Writing – review & editing

References

[1] H. Hanke and D. Knees, A phase-field damage model based on evolving microstructure, *Asymptotic Analysis* **101** (2017), 149–180.
[2] E. Lefever, A hybrid approach to domain-independent taxonomy learning, *Applied Ontology* **11**(3) (2016), 255–278.
[3] P.S. Meltzer, A. Kallioniemi and J.M. Trent, Chromosome alterations in human solid tumors, in: *The Genetic Basis of Human Cancer*, B. Vogelstein and K.W. Kinzler, eds, McGraw-Hill, New York, 2002, pp. 93–113.
[4] P.R. Murray, K.S. Rosenthal, G.S. Kobayashi and M.A. Pfaller, *Medical Microbiology*, 4th edn, Mosby, St. Louis, 2002.
[5] E. Wilson, Active vibration analysis of thin-walled beams, PhD thesis, University of Virginia, 1991.

Appendix A. Formalizing RO-Crate in First Order Logic

Below is a formalization of the concept of RO-Crate as a set of relations using First Order Logic:

A.1. Language

Definition of language $\mathcal{L}_{rocrate}$:

$$\mathcal{L}_{rocrate} = \{Property(p), Class(c), Value(x), \mathbb{R}, \mathbb{S}\}$$
$$\mathbb{D} = \mathbb{IRI}$$
$$\mathbb{IRI} \equiv \text{IRIs as defined in RFC3987}$$

$$\mathbb{R} \equiv \text{real or integer numbers}$$
$$\mathbb{S} \equiv \text{literal strings}$$

The domain of discourse $\mathbb{D}$ is the set of $\mathbb{IRI}$ identifiers [**?**] (notation `<http://example.com/>`)[64], with additional descriptions using numbers $\mathbb{R}$ (notation 13.37) and literal strings $\mathbb{S}$ (notation "Hello").

From this formalised language $\mathcal{L}_{rocrate}$ we can interpret an RO-Crate in any representation that can gather these descriptions, their properties, classes, and literal attributes.

A.2. Minimal RO-Crate

Below we use $\mathcal{L}_{rocrate}$ to define a minimal[65] RO-Crate:

$$\begin{aligned}
ROCrate(R) \vDash\ & Root(R) \land Mentions(R,R) \land hasPart(R,d) \land \\
& Mentions(R,d) \land DataEntity(d) \land \\
& Mentions(R,c) \land ContextualEntity(c) \\
\forall r\ Root(r) \Rightarrow\ & Dataset(r) \land name(r,n) \land description(r,d) \land \\
& published(r,date) \land license(e,l) \\
\forall e \forall n\ name(e,n) \Rightarrow\ & Value(n) \\
\forall e \forall s\ description(e,s) \Rightarrow\ & Value(s) \\
\forall e \forall d\ datePublished(e,d) \Rightarrow\ & Value(d) \\
\forall e \forall l\ license(e,l) \Rightarrow\ & ContextualEntity(l) \\
DataEntity(e) \equiv\ & File(e) \oplus Dataset(e) \\
Entity(e) \equiv\ & DataEntity(e) \lor ContextualEntity(e) \\
\forall e\ Entity(e) \Rightarrow\ & Class(e) \\
Mentions(R,s) \vDash\ & Relation(s,p,e) \oplus Attribute(s,p,l) \\
Relation(s,p,o) \vDash\ & Entity(s) \land Property(p) \land Entity(o) \\
Attribute(s,p,v) \vDash\ & Entity(s) \land Property(p) \land Value(v) \\
Value(v) \equiv\ & v \in \mathbb{R} \oplus v \in \mathbb{S}
\end{aligned}$$

An $ROCrate(R)$ is defined as a self-described *Root Data Entity*, which describes and contains parts (*data entities*), which are further described in *contextual entities*. These terms align with their use in the RO-Crate 1.1 terminology.

[64]For simplicity, blank nodes are not included in this formalisation, as RO-Crate recommends the use of IRI identifiers: https://www.researchobject.org/ro-crate/1.1/appendix/jsonld.html#describing-entities-in-json-ld

[65]The full list of types, relations and attribute properties from the RO-Crate specification are not included. Examples shown include $datePublished$, $CreativeWork$ and $name$.

The $Root(r)$ is a type of $Dataset(r)$, and must have the metadata to literal attributes to provide a $name$, $description$ and $datePublished$, as well as a contextual entity identifying its license. These predicates correspond to the RO-Crate 1.1 requirements for the root data entity.

The concept of an $Entity(e)$ is introduced as being either a $DataEntity(e)$, a $ContextualEntity(e)$, or both; and must be typed with at least one $Class(e)$.

For simplicity in this formalization (and to assist production rules below) R is a constant representing a single RO-Crate, typically written to independent RO-Crate Metadata files. R is used by $Mentions(R, e)$ to indicate that e is an Entity described by the RO-Crate and therefore its metadata (a set of $Relation$ and $Attribute$ predicates) form part of the RO-Crate serialization. $Relation(s, p, o)$ and $Attribute(s, p, x)$ are defined as a *subject—predicate—object* triple pattern from an $Entity(s)$ using a $Property(p)$ to either another $Entity(o)$ or a $Value(x)$ value.

A.3. Example of formalized RO-Crate

The below is an example RO-Crate represented using the above formalisation, assuming a base URI of `<http://example.com/ro/123/>`:

$ROCrate$(`<http://example.com/ro/123/>`)
$name$(`<http://example.com/ro/123/>`,
  "Data files associated with the manuscript:Effects of ...")
$description$(`<http://example.com/ro/123/`,
  "Palliative care planning for nursing home residents ...")
$license$(`<http://example.com/ro/123/>`,
  `<https://creativecommons.org/licenses/by-nc-sa/3.0/au/>`)
$datePublished$(`<http://example.com/ro/123/>`, "2017-02-23")
$hasPart$(`<http://example.com/ro/123/>`, `<http://example.com/ro/123/file.txt>`)
$hasPart$(`<http://example.com/ro/123/>`, `<http://example.com/ro/123/interviews/>`)

$ContextualEntity$(`<https://creativecommons.org/licenses/by-nc-sa/3.0/au/>`)
$name$(`<https://creativecommons.org/licenses/by-nc-sa/3.0/au/>`,
  "Attribution-NonCommercial-ShareAlike 3.0 Australia (CC BY-NC-SA 3.0 AU)")

$File$(`<http://example.com/ro/123/survey.csv>`)
$name$(`<http://example.com/ro/123/survey.csv>`, "Survey of care providers")

$Dataset$(`<http://example.com/ro/123/interviews/>`)

$$name(\texttt{<http://example.com/ro/123/interviews/>}, \text{“Audio recordings of care provider interviews”})$$

In reality many additional attributes from schema.org types like http://schema.org/Dataset and http://schema.org/CreativeWork would be used to further describe the RO-Crate and its entities, but as these are optional they do not form part of this formalisation.

A.4. Mapping to RDF with schema.org

A formalized RO-Crate in $\mathcal{L}_{rocrate}$ can be mapped to different serializations. Assume a simplified[66] language $\mathcal{L}_{RDF}$ based on the RDF abstract syntax:

$$\begin{aligned}
\mathcal{L}_{RDF} &\equiv \{Triple(s,p,o), IRI(i), BlankNode(b), Literal(s), \mathbb{IRI}, \mathbb{S}, \mathbb{R}\} \\
\mathbb{D}_{RDF} &\equiv \mathbb{S} \\
\forall i\, IRI(i) &\Rightarrow i \in \mathbb{IRI} \\
\forall s \forall p \forall o\, Triple(s,p,o) &\Rightarrow \big(IRI(s) \lor BlankNode(s)\big) \land \\
&\quad IRI(p) \land \\
&\quad \big(IRI(o) \lor BlankNode(o) \lor Literal(o)\big) \\
Literal(v) &\models Value(v) \land Datatype(v,t) \land IRI(t) \\
\forall v\, Value(v) &\Rightarrow v \in \mathbb{S} \\
LanguageTag(v,l) &\equiv Datatype\big(v, \\
&\quad \texttt{<http://www.w3.org/1999/02/22-rdf-syntax-nslangString>}\big)
\end{aligned}$$

Below follows a mapping from $\mathcal{L}_{rocrate}$ to $\mathcal{L}_{RDF}$ using schema.org as vocabulary:

$$\begin{aligned}
Property(p) &\Rightarrow type(p, \texttt{<http://www.w3.org/2000/01/rdf-schemaProperty>}) \\
Class(c) &\Rightarrow type(c, \texttt{<http://www.w3.org/2000/01/rdf-schemaClass>}) \\
Dataset(d) &\Rightarrow type(d, \texttt{<http://schema.org/Dataset>}) \\
File(f) &\Rightarrow type(f, \texttt{<http://schema.org/MediaObject>}) \\
CreativeWork(e) &\Rightarrow ContextualEntity(e) \land type(e, \texttt{<http://schema.org/CreativeWork>}) \\
hasPart(e,t) &\Rightarrow Relation(e, \texttt{<http://schema.org/hasPart>}, t) \\
name(e,n) &\Rightarrow Attribute(e, \texttt{<http://schema.org/name>}, n) \\
description(e,s) &\Rightarrow Attribute(e, \texttt{<http://schema.org/description>}, s)
\end{aligned}$$

[66]This simplification does not cover the extensive list of literal datatypes built-in to RDF 1.1, only strings and decimal real numbers. Likewise, language of literals are not included.

$$
\begin{aligned}
datePublished(e,d) &\Rightarrow Attribute(e, \texttt{<http://schema.org/datePublished>}, d)\\
license(e,l) &\Rightarrow Relation(e, \texttt{<http://schema.org/license>}, l) \land CreativeWork(l)\\
type(e,t) &\Rightarrow Relation(e, \texttt{<http://www.w3.org/1999/02/22-rdf-syntax-nstype>}, t)\\
&\land Class(t)\\
String(s) &\equiv Value(s) \land s \in \mathbb{S}\\
String(s) &\Rightarrow Datatype(s, \texttt{<http://www.w3.org/2001/XMLSchemastring>})\\
Decimal(d) &\equiv Value(d) \land d \in \mathbb{R}\\
Decimal(d) &\Rightarrow Datatype(d, \texttt{<http://www.w3.org/2001/XMLSchemadecimal>})\\
Relation(s,p,o) &\Rightarrow Triple(s,p,o) \land IRI(s) \land IRI(o)\\
Attribute(s,p,o) &\Rightarrow Triple(s,p,o) \land IRI(s) \land Literal(o)
\end{aligned}
$$

Note that in the JSON-LD serialization of RO-Crate, the expression of $Class$ and $Property$ is typically indirect: The JSON-LD `@context` maps to schema.org IRIs, which, when resolved as Linked Data, embed their formal definition as RDFa.

A.5. RO-Crate 1.1 Metadata File Descriptor

An important RO-Crate principle is that of being **self-describing** Therefore the serialisation of the RO-Crate into a file should also describe itself in a Metadata File Descriptor, indicating it is *about* (describing) the RO-Crate root data entity, and that it $conformsTo$ a particular version of the RO-Crate specification:

$$
\begin{aligned}
about(s,o) &\Rightarrow Relation(s, \texttt{<http://schema.org/about>}, o)\\
conformsTo(s,o) &\Rightarrow Relation(s, \texttt{<http://purl.org/dc/terms/conformsTo>}, o)\\
MetadataFile(m) &\Rightarrow CreativeWork(m) \land about(m,R) \land ROCrate(R) \land\\
&\quad conformsTo(m, \texttt{<https://w3id.org/ro/crate/1.1>})
\end{aligned}
$$

Note that although the metadata file necessarily is an *information resource* written to disk or served over the network (e.g. as JSON-LD), it is not considered to be a contained *part* of the RO-Crate in the form of a *data entity*, rather it is described only as a *contextual entity*.

While in the conceptual model the *RO-Crate Metadata File* can be seen as the top-level node that describes the *RO-Crate Root*, in the formal model (and the JSON-LD format) the metadata file descriptor is an additional contextual entity and not affecting the depth-limit of the RO-Crate.

A.6. Forward-chained Production Rules for JSON-LD

Combining the above predicates and schema.org mapping with rudimentary JSON templates, these forward-chaining production rules can output JSON-LD according to the RO-Crate 1.1 specification[67]:

$$
\begin{aligned}
Mentions(R,s) \wedge Relation(s,p,o) &\Rightarrow Mentions(R,o) \\
IRI(i) &\Rightarrow \texttt{"}i\texttt{"} \\
Decimal(d) &\Rightarrow d \\
String(s) &\Rightarrow \texttt{"}s\texttt{"} \\
\forall e \forall t\ type(e,t) &\Rightarrow \{\texttt{"@id"}:e, \\
&\quad \texttt{"@type"}:t \\
&\} \\
\forall s \forall p \forall o\ Relation(s,p,o) &\Rightarrow \{\texttt{"@id"}:s, \\
&\quad p\texttt{: \{ "@id"}:o\} \\
&\} \\
\forall s \forall p \forall v\ Attribute(s,p,v) &\Rightarrow \{\texttt{"@id"}:s, \\
&\quad p:v \\
&\} \\
\forall r \forall c ROCrate(r) &\Rightarrow \{\ \texttt{"@graph": [} \\
&\quad\ Mentions(r,c) * \\
&\quad \texttt{]} \\
&\} \\
R &\equiv \texttt{<./>} \\
R &\Rightarrow MetadataFile(\texttt{<ro-crate-metadata.json>})
\end{aligned}
$$

This exposes the first order logic domain of discourse of IRIs, with rational numbers and strings as their corresponding JSON-LD representation. These production rules first grow the graph of R

[67]**Limitations:** Contextual entities not related from the RO-Crate (e.g. using inverse relations to a data entity) would not be covered by the single direction $Mentions(R,s)$ production rule; see issue 122. The $datePublished(e,d)$ rule do not include syntax checks for the ISO 8601 datetime format. Compared with RO-Crate examples, this generated JSON-LD does not use a $@context$ as the IRIs are produced unshortened, a post-step could do JSON-LD Flattening with a versioned RO-Crate context. The `@type` expansion is included for clarity, even though this is also implied by the $type(e,t)$ expansion to $Relation(e, \texttt{xsd:type})$.

by adding a transitive rule – anything described in R which is related to o, means that o is also mentioned by the $ROCrate(R)$. For simplicity this rule is one-way; in practice the graph can also contain free-standing contextual entities that have outgoing relations to data- and contextual entities.

Appendix B. RO-Crate Community

As of 2021-05-24, the *RO-Crate* Community members are:

- Peter Sefton https://orcid.org/0000-0002-3545-944X (co-chair)
- Stian Soiland-Reyes https://orcid.org/0000-0001-9842-9718 (co-chair)
- Eoghan Ó Carragáin https://orcid.org/0000-0001-8131-2150 (emeritus chair)
- Oscar Corcho https://orcid.org/0000-0002-9260-0753
- Daniel Garijo https://orcid.org/0000-0003-0454-7145
- Raul Palma https://orcid.org/0000-0003-4289-4922
- Frederik Coppens https://orcid.org/0000-0001-6565-5145
- Carole Goble https://orcid.org/0000-0003-1219-2137
- José María Fernández https://orcid.org/0000-0002-4806-5140
- Kyle Chard https://orcid.org/0000-0002-7370-4805
- Jose Manuel Gomez-Perez https://orcid.org/0000-0002-5491-6431
- Michael R Crusoe https://orcid.org/0000-0002-2961-9670
- Ignacio Eguinoa https://orcid.org/0000-0002-6190-122X
- Nick Juty https://orcid.org/0000-0002-2036-8350
- Kristi Holmes https://orcid.org/0000-0001-8420-5254
- Jason A. Clark https://orcid.org/0000-0002-3588-6257
- Salvador Capella-Gutierrez https://orcid.org/0000-0002-0309-604X
- Alasdair J. G. Gray https://orcid.org/0000-0002-5711-4872
- Stuart Owen https://orcid.org/0000-0003-2130-0865
- Alan R Williams https://orcid.org/0000-0003-3156-2105
- Giacomo Tartari https://orcid.org/0000-0003-1130-2154
- Finn Bacall https://orcid.org/0000-0002-0048-3300
- Thomas Thelen https://orcid.org/0000-0002-1756-2128
- Hervé Ménager https://orcid.org/0000-0002-7552-1009
- Laura Rodríguez-Navas https://orcid.org/0000-0003-4929-1219
- Paul Walk https://orcid.org/0000-0003-1541-5631
- brandon whitehead https://orcid.org/0000-0002-0337-8610
- Mark Wilkinson https://orcid.org/0000-0001-6960-357X
- Paul Groth https://orcid.org/0000-0003-0183-6910
- Erich Bremer https://orcid.org/0000-0003-0223-1059
- LJ Garcia Castro https://orcid.org/0000-0003-3986-0510
- Karl Sebby https://orcid.org/0000-0001-6022-9825
- Alexander Kanitz https://orcid.org/0000-0002-3468-0652
- Ana Trisovic https://orcid.org/0000-0003-1991-0533
- Gavin Kennedy https://orcid.org/0000-0003-3910-0474
- Mark Graves https://orcid.org/0000-0003-3486-8193
- Jasper Koehorst https://orcid.org/0000-0001-8172-8981

- Simone Leo https://orcid.org/0000-0001-8271-5429
- Marc Portier https://orcid.org/0000-0002-9648-6484
- Paul Brack https://orcid.org/0000-0002-5432-2748
- Milan Ojsteršek https://orcid.org/0000-0003-1743-8300
- Bert Droesbeke https://orcid.org/0000-0003-0522-5674
- Chenxu Niu https://github.com/UstcChenxu
- Kosuke Tanabe https://orcid.org/0000-0002-9986-7223
- Tomasz Miksa https://orcid.org/0000-0002-4929-7875
- Marco La Rosa https://orcid.org/0000-0001-5383-6993