The OpenAIRE Research Graph Data Model

Version: 1.3 **Release date:** 17th of April, 2019

Authors: Paolo Manghi, Alessia Bardi, Claudio Atzori, Miriam Baglioni, Natalia Manola, Jochen Schirrwagen, Pedro Principe

Contributors: Michele Artini, Amelie Becker, Michele De Bonis, Nikos Houssos, Katerina Iatropoulou, Antonis Lempesis, Aenne Loeden, Stefania Martziou, Eloy Rodrigues

1 Institute of information science and technologies - CNR, Pisa, Italy
2 Interdisciplinary Centre for Mathematical and Computational Modelling - Warsaw, Poland
3 University of Minho - Minho, Portugal
4 BIelefeld University - Bielefeld, Germany
5 National Documentation Centre (EKT) - Athens, Greece
6 Athena Research & Innovation Center - Athens, Greece

Summary The purpose of the European OpenAIRE infrastructure is to facilitate, foster, support, and monitor Open Science scholarly communication in Europe. The infrastructure has been operational for almost a decade and successful in linking people, ideas and resources in support of the free flow, access, sharing, and re-use of research outcomes. To this aim it offers dissemination and training on Open Access and Open Science, facilitates exchange of knowledge, and operates the technical services required to facilitate and monitor Open Science publishing trends and research impact across geographic and discipline boundaries. OpenAIRE services populate a research graph whose objects are scientific results, organizations, funders, communities, organizations, and data sources. In this article we describe the data model, inspired by several existing metadata standards.

Table of Content

Introduction

The purpose of the European OpenAIRE infrastructure is to facilitate, foster, support, and monitor Open Science scholarly communication in Europe. The infrastructure has been operational for almost a decade and successful in linking people, ideas and resources in support of the free flow, access, sharing, and re-use of research outcomes. On the one hand, OpenAIRE manages and enables an open and participatory network of people willing to identify the commons and forums required to foster and implement OpenScience policies and practices in Europe and globally. On the other hand, it supports the technical services required to facilitate and monitor Open Science publishing trends and research impact across geographic and discipline boundaries. Such services collect bibliographic citation metadata from various Internet sources (e.g. libraries, publication repositories, publishers, author directories) and populate a research graph whose objects are scientific results, organizations, funders, communities, organizations, and data sources.

OpenAIRE started its mission back in 2009 focusing on monitoring the uptake of the EC Open Access mandes for publications. Over the years, the aim of the project followed the trends of the the research environment to naturally shift towards Open Science. Open Science scholarly communication grounds on transparency and reproducibility of science, which cannot be achieved without publishing all results of science. For such a reason, the OpenAIRE data model was updated to introduce the notion of research community and research datasets, research software, and “other products”. As a result, OpenAIRE can populate a research graph enabling monitoring of Open Science evolution, research trends, and research impact, for funders and organization. In this article we describe the current data model, inspired by several existing metadata standards, and how it matches this vision.

Rationale and background

The OpenAIRE data model describes structure and semantics of a research graph intended to:

- Enable citation and interlinking of scientific results across disciplines;
- Provide funders with statistics about the uptake of their Open Access mandates;
- Support research communities at publishing, interlinking, and discovering all kinds of scientific results in respect of their community practices (e.g. vocabularies);
- Reporting research impact and research trends for funders and institutions;
- Keep track of provenance of each object, property or relationship in the graph (e.g. data sources from which information is collected, algorithms inferring the information, system users providing the information).

The model addresses all such requirements and takes inspiration from two main initiatives: the CERIF data model and the OpenAIRE guidelines. In the following we present what they are and how they contributed to the model.

CERIF influence

Due to its CRIS-like nature, the OpenAIRE data model took inspiration from the CERIF data model.

CERIF[1] is a conceptual model of the research domain, typically applied in Common Research Information Systems (CRIS). It captures research results (publications, patents, products – the latter covering datasets, software and other types of output) as well as entities constituting the research context, like persons, organizations, projects, funding programmes, facilities, services.

Among several others, a key feature of CERIF is the ability to represent semantic relationships (e.g. person-publication, organization-project, project-funding programme). Relationships in CERIF are called *link entities* and contain temporal information specifying the date range within which a specific semantic relationship applies, for example person A was coordinator of project X between 01-Feb-2012 to 29-Jun-2012. The semantics of each relationship instance (e.g. the role of a person in a project) and the associated vocabularies (i.e. potential values for roles) are not static components of the CERIF entity structure, but can be dynamically injected into a CERIF database. This is accomplished using the *CERIF Semantic Layer*, which enables the specification and maintenance of controlled vocabularies, called classification *schemes*, and their terms, called *classes*, as well as their association with entities. CERIF is able to represent any vocabulary structure (e.g. thesaurus) and the mapping among terms in different vocabularies. The semantic layer is also used to directly represent classifications of CERIF entities, example.g. terms from a subject classification vocabulary can be assigned to a publication, organisations can be typed. While a CERIF-based system is extensible to include any vocabulary, a set of common vocabularies is published as a separate component of the CERIF standard. The design and structure of the semantic layer facilitates the generation of Linked Open data from CERIF databases, which is being standardized by the Linked Open Data Task Group of euroCRIS.

The OpenAIRE data model has adopted linked entities as a tool to flexibly include new semantics without updating the model, and therefore the applications operating over such model. Other features were instead disregarded for the moment (e.g. multilinguality, multiple funders for projects) while finding a balance between development and maintenance cost, ability to collect information matching the model (over-representation), matching application requirements, and the usual system delivery deadlines.

OpenAIRE Guidelines

The description, citation and indexing of scientific results require a consistent use of metadata schemas that make scientific results findable and accessible across infrastructure boundaries, ensuring sharing and reuse of research in a transparent and reliable manner. The OpenAIRE Guidelines for Content Providers describe different types of research artifacts, specifying basic metadata fields, vocabularies, protocols, and illustrating best practices that come from the library world, the research data world, and the world of research software, while covering the recent trends in research infrastructures. They are well fitted to bridge the interoperability gaps between research infrastructures at large, their implementation being the stepping-stone for a linked data infrastructure for research, providing a 360º contextual view of research. In particular, OpenAIRE offers five different sets of Guidelines for content providers covering a variety of research repository classes:

- The Guidelines for Institutional and Thematic Repositories and Publishing Platforms are based primarily on the description of research literature. They are intended to guide repository managers to expose metadata of all scientific publications including, whenever applicable, references to research projects and EU or other funders.
- The Guidelines for Data Archive Managers are based on the description of research data. They use the well-established DataCite metadata schema, provide extensions of metadata properties and controlled vocabularies where necessary and, describe how to interpret these metadata

[1] https://www.eurocris.org/cerif/main-features-cerif

properties according to the nature of the respective research product. They specifically facilitate the linking of research data to publications, software and other research products.

- The Guidelines for Software Repository Managers provide orientation for the description of software products. These give immediate visibility of software as a "citable research product", having been defined with a pragmatic approach, keeping mandatory properties to the minimum, focusing on properties for citation (attribution and access), and having the possibility for future addition of properties, while disregarding discover-for-reuse properties. Such set of guidelines has been inspired by several initiatives and/or endorsed by the following initiatives" Force 11 Software Citation Principles, DataCite, OpenMinTed SHARE-OMTD, Codemeta initiative, DOE CODE initiative.
- The Guidelines for Other Research Products (ORPs) focus on the description of research products different from literature, data and software such as research services, protocols, workflows, methods, virtual appliances etc. provided they are available in open access or linked to a publication, project or research community in OpenAIRE.
- The Guidelines for CRIS Managers are designed to map comprehensive research information using the CERIF standard. By implementing then, CRIS managers support the inclusion and therefore the reuse of metadata in their systems within the OpenAIRE infrastructure. For developers of CRIS platforms, the Guidelines provide guidance to add supportive functionalities for CRIS managers and users. Finally, it is worth noting that exchange of information between individual CRIS systems and the OpenAIRE infrastructure is an example of point-to-point data exchange between CRIS systems, since the OpenAIRE infrastructure is itself a CRIS system.

Since we expect the data model to represent all properties that are requested to the data sources of OpenAIRE, entities in the model match the guideline properties. Literature, software, datasets, and ORPs results have common properties represented in the Result entity and the specific ones in the relative sub-types.

The Data Model

General Concepts

The main entities of the OpenAIRE information space are: research results (datasets, literature, software and other types of research products), organisations, funders, funding streams, projects, and data sources (aka content providers).

Research results represent outcome of research activities that may be funded via a research project. OpenAIRE initially proposed two kinds of results: *datasets* (e.g., experimental data, software products) and *publications*. Based on common practices of scientists and to foster the adoption of Open Science principles, OpenAIRE included research *software* as a research result and a generic *other research product* (ORP) category for different types of research outputs. It is worth noticing that scientific products are intended to evolve: whenever a sub-category of ORP shows to have a critical mass in terms of usage, the OpenAIRE data curation board may decide to elect it as a "first-class citizen" of the model, at the same league of publications, datasets, and software.

In the real world, research results have typically different "manifestations" (i.e. physical representations). For example, the same publication may be deposited by its authors in different repositories, exposing payload files (e.g., PDF) and with different access right (e.g. OA and non-OA). OpenAIRE applies deduplication to detect such duplicates and create one result representing their merge. Each manifestation is called instance, is associated with the original data source, and is part of the representative result. The result will contain details of the instances and also include the merge of the instance properties to build a potentially richer record.

Organizations include companies, research centers or institutions involved as project partners or as

responsible of operating data sources. Information about organizations are collected from funder databases like CORDA, registries of data sources like OpenDOAR and re3Data, and CRIS systems.
Of crucial interest to OpenAIRE is also the identification of the funders (e.g. European Commission, WellcomeTrust, FCT Portugal, NWO The Netherlands) that co-funded the projects that have led to a given result. Funders can be associated to a list of funding streams (e.g. FP7 and H2020 for the EC), which identify the strands of fundings comprised by the funding stream. Funding streams can be nested to form a tree of sub-funding streams, and projects are typically associated to the funding stream “leaves” of such trees.

Finally, OpenAIRE entity instances are created out of data collected from various data sources of different kinds, such as publication repositories, dataset archives, CRIS systems, etc. Data sources export information packages (e.g., XML records, HTTP responses, RDF data, JSON) that may contain information on one or more of such entities and possibly relationships between them. It is important, once each piece of information is extracted from such packages and inserted into the information space as an entity, for such pieces to keep provenance information relative to the originating data source. This is to give visibility to the data source, but also to enable the reconstruction of the very same piece of information if problems arise.

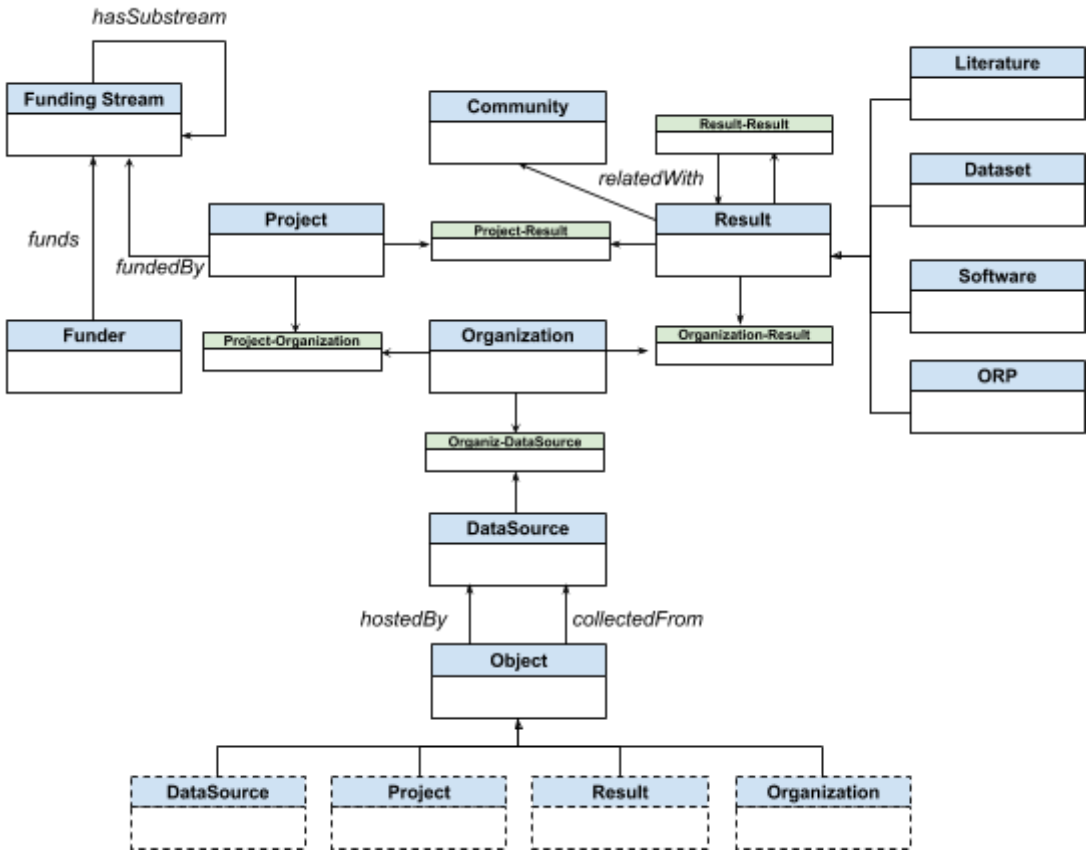

Figure 2 OpenAIRE Data Model.

OpenAIRE and the CERIF semantic layer

According to the CERIF's data model vision: (i) “horizontal” classification of entities (e.g., by vocabularies of terms) is not modeled through properties associated to given controlled vocabularies and (ii) semantic relationships between entities are not modeled by adding dedicated relationships. In both cases, CERIF introduces a flexible modeling mechanism which allows injecting classification semantics into “semantics-agnostic” entities and relationships. The mechanism is obtained by

introducing two entities Schemes and Classes such that:

- *Class* A Class represents one term of a classification, e.g., vocabulary, taxonomy, under a given Scheme. As such it is characterized by the following properties: a Code, which represents the persistent identifier associated to the term (e.g., real-world classifications, such as ISO vocabularies for countries, have a standard identification code for terms), a name, an acronym, a description, a StartDate, and an EndDate.
- *Scheme* A Scheme identifies the existence of a classification scheme, which is modeled as a set of Class objects. A Scheme is characterized by the following properties: a Code, which represents the persistent identifier associated to the Scheme (e.g., real-world schemes, such as taxonomies, may have a standard identification code), a name, an acronym, a description, a StartDate, and an EndDate.

According to the CERIF's definition, Classes and Schemes can be themselves interlinked to form arbitrary complex lattices of Classes and Schemes, respectively. *In OpenAIRE we adopt a lighter interpretation*, by introducing the pair Scheme/Class whenever we need to introduce a property of type *Qualifier*, i.e. a property whose value comes from a controlled vocabulary, or a relationship between core entities in the model. Such mechanisms allow to flexibly inject relationship semantics and vocabularies into the data model.

OpenAIRE entities, relationships and types

The entities in the data model belong to the following categories:

- **Core entities**: the entities whose information is continuously and incrementally fed to the information space and is of interest to OpenAIRE end-users; namely **Result** (Literature, Dataset, Software, Other products), **Organization**, **DataSource**, **Projects**, **Funder, Funding Stream**;
- **Linking entities**: entities used to model relationships, used to connect in a semantic-agnostic way two or more main entities; namely, those denoted by an Entity1_Entity2 notation (see aforementioned CERIF semantic layer).
- **Types**: types are used to define structured values for entity properties. Structured values differ from objects in the sense they do not have an identity, i.e. cannot be referred to by relationships from other objects.

Core Entities

In this section the core entities of the data model are introduced by describing the concept or real-world entity they represent and introducing their descriptive properties and relationships with other entities.

Results are intended as digital objects, described by metadata, resulting from a scientific process. Its sub-entity types are Literature, Dataset, Software, and Other Research Product, which inherit all Result properties and relationships with other entities and add their specific ones:

- **Literature** includes all digital research artefacts whose intended use is narrative storytelling of a research activity and its results. Examples are scientific articles, reports, slides, data papers, etc. Although there are exceptions, as each scientist has a large degree of freedom in publishing and interlinking his artefacts, it can be generally assumed that literature artefacts are published with a narrative intent. For those specific cases where literature is intended for different use, we in general do not expect scientists to publish such artefacts as literature artefacts. For example when an article is a carrier of readable datasets (e.g. articles with tables) the article is often deposited a second time in a data repository, assigned a new DOI,

and marked as a dataset of type “textual”; in the case articles full-texts are used for natural language processing (NLP), scientists will likely create a dataset of type “collection of articles”.

- **Datasets** include digital research artefacts encoding experimental or real-world observations/measures (e.g. primary data), secondary data derived from programmatic processing of other datasets, or more generally digital representations of facts to be interpreted by a program. The definition is cross-discipline, hence spans across multiple interpretations of datasets, where typologies and granularity obey to different scientific facets. Examples include, but are not limited to: databases (e.g. Worms), records of databases (e.g. proteins in the UniProt database), table files, queries over databases (time-series slices, geospatial maps, SQL queries), media (e.g. images, videos) or collections of media.
- **Software** entities represent research software, i.e. software that is an output of a research activity. Examples include, but are not limited to: code scripts, web services and web applications.
- **Other research products** include any research output that is not literature, data, or software. Examples include, but are not limited to: algorithms, scientific workflows/pipelines, protocols, standard operating procedure (SOP), simulations, mathematical and statistical models, but also research packages. Research packages can group a set of research artefacts, but can also include the encoding of a composition logic that binds them together. For example, an instance of a workflow is a package that describes the combination of specific artefacts to implement a scientific process, execute an experiment, etc.

Communities Communities are intended as groups of people with a common research intent and can be of two types: *research initiatives* or *research communities*. The former are intended to capture a view of the information space that is “research impact”-oriented, i.e. all products generated due to my research initiative, the latter “research activity” oriented, i.e. all products that may be of interest or related with my research initiative. For example, the organizations supporting a research infrastructure fall in the first category, while the researchers involved in a discipline fall in the second.

Organizations include companies, research centers or institutions involved as project partners or as responsible of operating data sources. Information about organizations are collected from funder databases like CORDA, registries of data sources like OpenDOAR and re3Data, and CRIS systems, as being related to projects or data sources.

Funders, funding streams and projects. Of crucial interest to OpenAIRE is also the identification of the funders (e.g. European Commission, WellcomeTrust, FCT Portugal, NWO The Netherlands) that co-funded the projects that have led to a given result. Funders can be associated to a list of funding streams (e.g. FP7, H2020 for the EC), which identify the strands of fundings. Funding streams can be nested to form a tree of sub-funding streams. Projects are typically associated to the funding stream “leaves” of such trees.

Data sources Finally, OpenAIRE entity instances are created out of data collected from various data sources of different kinds, such as publication repositories, dataset archives, CRIS systems, funder databases, etc. Data sources export information packages (e.g., XML records, HTTP responses, RDF data, JSON) that may contain information on one or more of such entities and possibly relationships between them. For example, a metadata record about a project carries information for the creation of a Project entity and its participants (as Organization entities). It is important, once each piece of information is extracted from such packages and inserted into the OpenAIRE information space as an entity, for such pieces to keep provenance information relative to the originating data source. This is to give visibility to the data source, but also to enable the reconstruction of the very same piece of information if problems arise.

Objects Objects are all entities that can be collected form DataSources, for which we need to keep

provenance of collection.

Core entity: Object

Relationship	Target Type	Multiplicity	Description
CollectedFrom	Datasource	1..N	A Datasource from which metadata of this instance has been collected (e.g. an aggregator of institutional repositories)
HostedBy	Datasource	1..N	The data source that hosts and makes available the resources of this instance (e.g. an institutional repository)

The difference between collectedFrom and hostedBy relationships is introduced to model the concept of Aggregator data sources. Aggregator data sources are themselves aggregating information packages from a set of data sources; they differ from the data sources they aggregate, but play an equally important role in delivering the entities to OpenAIRE, and should therefore be given visibility. Accordingly, in order to guarantee visibility and ROI (Return Of Investment) to all data sources involved in this chain, when OpenAIRE collects information packages from Aggregators such packages are characterised by the data source from which they were “collected”, i.e. the aggregator itself, and the data sources where they were originally "hosted". Note that in the case of other typologies of data sources (e.g. repositories) *collectedFrom* and *hostedBy* refer to the same data source.

Core entity: Datasource (isA Object)

A Datasource is here intended as the (metadata) description/representation of a provider exporting (meta)data about scholarly communication results. Based on the typology of the provider and of the exported material, a Datasource is classified according to the categorization defined by the vocabulary dnet:datasource_typologies_vocabulary. Datasource objects can be registered to the system by users or their metadata be collected from Datasources of type "Entity Registry".

Property	Type	Multiplicity	Description
Type	Structured(-, Qualifier, -)	0..1	Typology of the datasource. For an updated list of possible typologies refer to the Qualifier.scheme dnet:datasource_typologies
OpenAIRE compatibility	Structured(-, Qualifier, -)	0..1	Compatibility of the datasource with respect to the OpenAIRE Guidelines. Qualifier.scheme is dnet:datasourceCompatibilityLevel
Official name	Structured(String,-,Provenance)	0..1	
English name	Structured(String,-,Provenance)	0..1	
Web site url	Structured(String,-,Provenance)	0..1	
Logo url	Structured(String,-,Provenance)	0..1	
Email contact	Structured(String,-,Provenance)	0..1	
Namespace Prefix	Structured(String,-,Provenance)	0..1	
Latitude	Structured(String,-,Provenance)	0..1	

Longitude	Structured(String,-,Provenance)	0..1	
Date of validation	Structured(Date,-,Provenance)	0..1	
Descritpion	Structured(String,-,Provenance)	0..1	
Subject	Structured(String,Qualifier,Provena nce)	1..N	the scientific discipline covered by the Datasource. The Qualifier part specifies the subject classification scheme the value belongs to. The list of supported subject classification schemes is defined in Qualifier.scheme dnet:subject_classification_typologies

Relationship	Target	Multiplicity	Description
responsibleOrganization	Datasource_Organization	0..1	The Organization related to this Datasource. For example, if the Datasource is an institutional repository, the institution operating the Datasource. Qualifier class and scheme is class="isResponsibleFor" scheme="dnet:datasource_organization_typologies"

The two major entity registries providing Datasources for literature repositories and data repositories are, respectively, OpenDOAR and re3data. Datasources collected from those two entity registries feature more detailed information (see subclasses *OpenDOAR Datasource* and *re3data Datasource*).

Sub-entity: OpenDOAR Datasource (isA DataSource)

An OpenDOAR Datasource is a Datasource collected from the OpenDOAR entity registry.
The following properties of an OpenDOAR Datasource are directly mapped from the information provided by OpenDOAR:

Property	Type	Multiplicity	Description
Number of items	Structured(String,-,Provenance)	0..1	
Date of number of items	Structured(String,-,Provenance)	0..1	
Policies	Structured(String,-,Provenance)	0..1	
Languages	Structured(String,-,Provenance)	0..1	
Content types	Structured(String,-,Provenance)	0..1	
Access info package	Structured(String,-,Provenance)	0..1	

Sub-entity: re3Data Datasource (isA Datasource)

A re3data Datasource is a Datasource collected from the re3data entity registry.
The following properties of a re3data Datasource are directly mapped from the information provided by re3data:

Property	Type	Multiplicity	Description

Release start date	Structured(String,-,Provenance)	0..1	
Release end date	Structured(String,-,Provenance)	0..1	
Mission statement url	Structured(String,-,Provenance)	0..1	
Data provider	Structured(Bool,-,Provenance)	0..1	
Service provider	Structured(Bool,-,Provenance)	0..1	
Database access type	Structured(String,-,Provenance)	0..1	
Database access restrictions	Structured(String,-,Provenance)	0..1	
Data upload type	Structured(String,-,Provenance)	0..1	
Data upload restricitions	Structured(String,-,Provenance)	0..1	
Versioning	Structured(Bool,-,Provenance)	0..1	
Citation guideline url	Structured(String,-,Provenance)	0..1	
Quality management kind	Structured(String,-,Provenance)	0..1	
PID systems	Structured(String,-,Provenance)	0..1	
Certificates	Structured(String,-,Provenance)	0..1	
Policies	Structured(Map,-,Provenance)	0..1	

Core entity: Organization (isA Object)

An Organization is here intended as the (metadata) description/representation of an organization involved in the scholarly communication chain, such as companies, research centers and institutions involved as project partners or as responsible of operating Datasources.
Information about organizations is collected from entity registry data sources providing information about projects and CRIS systems. Organizations collected from CORDA (datasource for the EC FP7 projects) belongs to a subclass of Organization: *CORDA Organization*.

Property	**Type**	**Multiplicity**	**Description**
persistentIdentifier	Structured(String,Qualifier,-)		
Legal short name	Structured(String,-,Provenance)	0..1	
Legal name	Structured(String,-,Provenance)	0..1	
Web site url	Structured(String,-,Provenance)	0..1	
Logo url	Structured(String,-,Provenance)	0..1	
Equivalent shortnames	Structured(String,-,Provenance)	0..N	
Equivalent names	Structured(String,-,Provenance)	0..N	

Relationship	**Target**	**Multiplicity**	**Description**

CollectedFrom	Data source	1..N	A Datasource from which this Organization has been collected (e.g. an aggregator of institutional repositories, an entity registry)
relatedDataSources	Datasource_Organization	0..N	The Datasources related to this Organization. For example, if an organization operates an institutional repositories, then the institutional repository is a Datasource related to the organization. Qualifier class and scheme is class="relatedDataSource" scheme="dnet:datasource_organization_typologies"
isParticipantOf	Organization_Project	0..N	The Projects this Organization is involved into. Qualifier class and scheme is class="isBeneficiaryOf" scheme="dnet:organization_project_typologies"
relatedResults	Organization_Result	0..N	The Results whose creators are affiliated with the Organization. Qualifier class and scheme is class="isRelatedWith" scheme="dnet:organization_result_typologies"

Core entity: Funder (isA Organization)

A Funder is here intended as the (metadata) description/representation of an Organization that provides research fundings.

Property	Type	Multiplicity	Description
Jurisdiction	Structured(String,Qualifier,-)	1..N	Jurisdictions of funders, e.g. the Commission, US for NSF, countries for National funders
PID	Structured(String,Qualifier,Provenance)	0..N	unique and persistent identifier used to identify the funder together with the relative identification agency, e.g. FundRef (not yet used in OpenAIRE)

Relationship	Target	Multiplicity	Description
Funds	FundingStream	1..N	Funding Stream of this Funder

Core entity: Funding Stream

Funding Streams identify the strands of fundings comprised by the funding stream. Funding streams can be nested to form a tree of sub-funding streams, and projects are typically associated to the funding stream “leaves” of such trees.

Constraint: in order to facilitate the construction of usable user interfaces, the hierarchy can be of at most three levels starting from the funder.

Property	Type	Multiplicity	Description
Identifier	String	1..1	
Name	String	1..1	
Description	String	1..1	

Relationship	Target	Multiplicity	Description
funded by	Funder	1..1	

has sub-stream	FundingStream	0..N	

Core entity: Project (isA Object)

A Project is here intended as the (metadata) description/representation of a research project.

Property	Type	Multiplicity	Description
Project code	Structured(String,-,Provenance)	1..1	
Title	Structured(String,-,Provenance)	1..1	
Acronym	Structured(String,-,Provenance)	0..1	
Call identifier	Structured(String,-,Provenance)	0..1	
Contract type	Structured(String,-,Provenance)	0..1	
Keywords	Structured(String,-,Provenance)	0..1	
Web site url	Structured(String,-,Provenance)	0..1	
Start date	Structured(Date,-,Provenance)	0..1	
End date	Structured(Date,-,Provenance)	0..1	
Duration	Structured(String,-,Provenance)	0..1	
EC SC39	Structured(Bool,-,Provenance)	0..1	This property is only available for EC funded projects
OA Mandate Publications	Structured(Bool,-,Provenance)	0..1	if EC SC39 exists, this property has the same value; for H2020 projects this property has value TRUE; default value FALSE
Subjects	Structured(-, Qualifier, -)	0..N	Set of subjects of the project. Controlled vocabulary to be specified soon.
EC article29-3 (**field to be confirmed, could be replaced with a generic *mandate* field**)	Structured(Bool,-,Provenance)	0..1	This property corresponds to the opt-in opt-out close of the EC Data Pilot; it is only available for EC H2020 funded projects, for others is set to FALSE

Relationship	Target	Multiplicity	Description
CollectedFrom	Data source	1..N	A Datasource from which this Project has been collected (e.g. an aggregator of institutional repositories, an entity registry)
Funded by	FundingStream	1..N	Funding Stream that funded this project
hasParticipants	Project_Organization	0..N	Organizations that are beneficiaries of the project. Qualifier class and scheme is class="beneficiaryOf" scheme="dnet:project_organization_typologies"
funds	Result_Project	0..N	Results that are co-funded by this project. Qualifier class and scheme is class="fundedResult" scheme="dnet:result_project_typologies"

Core entity: Result (isA Object)

A Result is here intended as the (metadata) description/representation of a scientific output (possibly). Results has the following sub-entities: Literature, Dataset, Software, and Other Products. Results can be the representation of the merge of a number of equivalent results.

Property	Type	Multiplicity	Description
Title	Structured(String,Qualifier,Provenance)	1..N	the titles of the Result, each with a typology represented by a Qualifier and Provenance information. Qualifier.scheme is dnet:dataCite_title.
Creator	Creator	0..N	The list of Creators of the Result
Date of acceptance	Date	1..1	
Publisher	Structured(String,-,Provenance)	0..1	
Description	Structured(String,-,Provenance)	0..N	Contains the Abstract of the Result
PID	Structured(String,Qualifier,Provenance)	0..N	unique and persistent identifier used to identify the result together with the relative identification agency. The identification agency is expressed in the Qualifier part. Qualifier.scheme is dnet:pid_types
Language	Structured(-, Qualifier, -)	0..1	the language used in the description or body of the Result, specified according to the ISO639 3-letter language codes. Qualifier.scheme is dnet:languages
Subject	Structured(String,Qualifier,Provenance)	0..N	the scientific discipline covered by the Result. The Qualifier part specifies the subject classification scheme the value belongs to. The list of supported subject classification schemes is defined in Qualifier.scheme dnet:subject_classification_typologies
Instance	Instance	1..N	An Instance of the Result, which represents a physical location where the Result files (web resources entities, possibly identified by DOIs) can be found

External Reference	ExternalReference	0..N	A link to an external resource that is not available as an entity in OpenAIRE. A reference is described by a web site name, a web URL, an identifier and the type of reference (e.g. a protein referred in a publication is represented as an external reference to the Protein Data Bank, with type 'accessionNumber')
Source	Structured(String,-,Provenance)	0..N	maps the dc:source element
Context	Context	0..N	Information about the fundings and/or research initiatives related to the Result
Country	Structured(-, Qualifier, -)	0..N	the countries of the organizations to which the authors of the Result are affiliated to at the moment of publishing; values specified according to the ISO3166 2-letter country codes. Qualifier.scheme is dnet:countries Countries of the Organizations that are affiliations for the result (values are kept in sync with the country value within Organizations related with the result). Countries in this field may also be obtained from the country associated with Funders or DataSources, which indirectly indicate the jurisdiction of the result.
Best Access Rights	Structured(-, Qualifier, -)	1..1	The most "Open" license value found among the Instance s of a Result. Qualifier.scheme is dnet:access_modes For Publications the value belongs to the vocabulary [1]

Relationship	**Target**	**Multiplicity**	**Description**
isFundedBy	Result_Project	0..N	Links to the Projects that co-funded the research underlying the Result. Qualifier class and scheme is class="isFundedBy" scheme="dnet:result_project_typologies
hasAmongTopNSimilarDocuments/isAmongTopNSimilarDocuments	Result_Result	0..N	Link to other Result entities considered similar in content (e.g. similar publications). Qualifier class and scheme is class="hasAmongTopNSimilarDocuments/isAmongTopNSimilarDocuments" scheme="dnet:result_result_typologies"
hasAuthorInstitution	Organization_Result	0..N	Links to one or more Institutions that are affiliations of the author of the result. The Qualifier class/scheme is: class="hasAuthorInstitution" scheme="dnet:result_organization_typologies"
aggregatedBy	Result_Result	0..N	Links to an ORP Result that "aggregates" the result. Qualifier class and scheme is class="isAggregatedBy" scheme="dnet:result_result_typologies"

relatedCommunities	Community	0..N	Link to Community entities to which the Result is relevant
isVersionOf	Result	0..N	Link to a Result that is another version of this Result

[1] Access right ordering: the Best Access Rights is automatically assigned as the "openest" license available among the licenses of the Instances of the Result based on the following ordering, from the "openest" to the "closest":
OPEN > 6MONTHS > 12MONTHS > EMBARGO > RESTRICTED > CLOSED > UNKNOWN

Sub-entity: Literature (isA Result)

Literature entities match Result entities. They will be extended to include journal properties.

Sub-entity: Datasets (isA Result)

Literature entities match Result entities.

Sub-entity: Software (isA Result)

Software entities represent research software, i.e. software that is an output of a research activity. Examples include, but are not limited to: code scripts, web services and web applications.
As subclass of Results, Software inherits all properties and relationships of the Result entity. In addition, they bear the specific properties in the Table below.

Property	Type	Multiplicity	Description
Contact person	Person	0..n	Information on the person responsible for providing further information regarding the resource
Contact group	String	0..n	Information on the group responsible for providing further information regarding the resource
Software type	String (for communities the value is selected from vocabulary specified by community)	1..1	Specifies the type of the software being described
Distribution Location	URL	0..N	URL of the web location from which the software can be directly downloaded
Documentation	URL	0..N	URL to a resource that provides useful information about the software to the end-users, such as execution tips, FAQs, help forums, etc.
Programming Language	String	0..n	Programming language in which the software is implemented.
Version	String	0..1	Version of the software
Tool	String	0..N	IT tool/service that can execute the software. E.g. if the software is an R script, then it can be executed by the tool "R Studio".
Distribution Form	String	0..N	The form in which the software is distributed. E.g. "source code", "executable", "zip".

Sub-entity: Other research product (isA Result)

The class of Results Other Research Products includes any research output that is not literature,

data, or software. Examples include, but are not limited to: algorithms, scientific workflows/pipelines, protocols, standard operating procedure (SOP), simulations, mathematical and statistical models, but also research packages. Research packages can group a set of research artefacts, but can also include the encoding of a composition logic that binds them together. Such logic must obey to specification or programming languages, agreed on across the community, and to be interpreted by humans or machines. For example, an instance of a workflow is a package as it describes the combination of specific artefacts to implement a scientific process, execute an experiment, etc. It is worth pointing out that, although strongly recommended and convenient for scientific publishing practices, artefacts and links specified within a representation of a research package are not necessarily represented as objects and links in the OpenAIRE research graph.
As subclass of Results, Other Research Product inherits all Result entity properties and relationships introduced above. In addition, they bear the specific properties and relationships specified below.

Property	Type	Multiplicity	Description
Contact person	Person	0..N	Information on the person responsible for providing further information regarding the resource
Contact group	String	0..N	Information on the group responsible for providing further information regarding the resource
Distribution Location	URL	0..N	URL of the web location from which the research product can be directly downloaded
Documentation	URL	0..N	URL to a resource that provides useful information about the product to the end-users.
Version	String	0..1	Version of the research product
Tool	String	0..N	IT tool/service that can execute the research product. E.g. if the product is a Taverna workflow it can be executed with the Taverna workbench.

Relationship	Target	Multiplicity	Description
Aggregates	Result_Result	0..N	Link the ORP to Result entities aggregated by the ORP, if any. Qualifier class and scheme is class="Aggregates" scheme="dnet:result_result_typologies"

Core entity: Communities

Communities are characterised by the properties and relationships in the Tables below. Several properties are kept outside of the model, as configuration parameters for the User Interfaces and the Research Community Dashboard application. They are listed here in a separate table for the sake of clarity and alignment with other documents.

Property	Type	Multiplicity	Description
Name	String	1	The name of the community

Relationship	Target	Multiplicity	Description
hasRelevantResults	Result	0..N	Results relevant to the community

Configuration properties	Description
relatedSubjects	
Controlled vocabularies for types and formats of software and other research product	
Inference parameters	
Monitoring parameters	
relevantProjects	Projects relevant to the community. Kept in configuration object
hasRelevantDatasource	Datasources whose content is relevant to the community. Each data source is accompanied by criteria of selection of the records.
hasRelevantZenodoCommunities	Id of Zenodo communities whose products are relevant for the community. Each Zenodo Community is accompanied by criteria of selection of the records.

Linking entities

In line with CERIF, linked entities are introduced to generalize on the notion of relationship between main entities of the model, in order to flexibly introduce new relationship semantics in the model without changing its structure. A linking entity explicitly models a relationship and describes it by means of a provenance property (e.g. when the relationship is inferred) and a Qualifier, to include the relationship semantics as a term of a controlled by dynamic list of terms (vocabulary).

Property	Type	Multiplicity	Description
Provenance	Provenance	1..1	
Qualifier	Qualifier	1..1	

In the following we shall introduce the vocabularies and terms currently available for the linked entities introduced by the model.

Linking entity: Result_Organization

Currently, only one Qualifier.Scheme is available for Result_Organization relationship named "dnet:results_organizations_typologies", whose values are:

- *hasAuthorInstitution/affiliatedResults*: an organization is the affiliation of authors of a result.

Linking entity: Datasource_Organization

Currently, only one Qualifier.Scheme is available for Datasource_Organization relationship named "dnet:datasources_organizations_typologies", whose values are:

- *responsibleOrganization/isResponsibleOf*: an organization is responsible for a Datasource (operation and sustainability)

Linking entity: Project_Organization

Currently, only one Qualifier.Scheme is available for Project_Organization relationship named "dnet:project_organization_typologies", whose values are:

- *isParticipantOf/hasParticipant*: an Organization is participant in a Project

Linking entity: Result_Project

Currently, only one Qualifier.Scheme is available for Result_Project relationship named "dnet:result_project_typologies", whose values are:

- *isFundedBy/funds*: the result has been co-funded by the project

Linking entity: Result_Result

Currently, only one Qualifier.Scheme is available for Result_Result relationship named "dnet:result_result_typologies", whose values are:

- *hasAmongTopNSimilarDocuments/isAmongTopNSimilarDocuments*: r1 hasAmongTopNSimilarDocuments r2 means that Results r1 and r2 are similar and that r2 isAmongTopNSimilarDocuments of r1; r1 isAmongTopNSimilarDocuments r2 means that Results r1 and r2 are similar and that r2 hasAmongTopNSimilarDocuments of r1;
- *isRelatedTo*: two results are somehow related to each other. OpenAIRE may further refine the semantics of possible types of "relatedness" by adding new classes in the Qualifier.scheme "dnet:result_result_typologies";
- other relationships will be included, such as *isCitedBy, isVersionOf, isAggregatedBy/Aggregates*.

Structured Types

Structured Type: Person

The Person type describes the structure of a person in the model, e.g. creators, project contacts.

Property	Type	Multiplicity	Description
Firstname	String	0..1	
Surname	String	0..1	
FullName	String	1..1	
Identifiers	Structured(String, Qualifier, -)	0..N	A list of identifiers (PIDs) such as ORCID, VIAF, available for the author. Each value is also qualified with the PID schema, provided via a vocabulary of PID schemas.

Structured Type: Creator

The Creator types describes a Person that is the creator of a Result and has a specific ranking, i.e. order of importance.

Property	Type	Multiplicity	Description
Person	Creator	1..1	
Rank	Number	0..1	
Affiliation	Organization	0..1	

Structured Type: Provenance

Provenance information is always associated to either a property value (e.g. subject property of a publication) or an object (e.g. a publication).

Property	Type	Multiplicity	Description
Inferred	Bool	1..1	TRUE if value is inferred, which includes generated by de-duplication process
DeletedByInference	Bool	1..1	TRUE if value is logically deleted by a process
Trust	Float(0,1)	1..1	Represents level of trust of the agent that generated value/object
Inference provenance	String	0..1	Free text specifying information about the agent that generated the value/object, e.g. algorithm name and version
ProvenanceAction	Qualifier	1..1	terms of this qualifier are to be taken from the following set [1]

[1] Provenance actions in OpenAIRE (currently not thoroughly applied across infrastructure services, needs rethinking and common agreement):

- sys:crosswalk:{data source typology}
- sys:iis:{algorithm name}
- sys:deduplication:{algorithm configuration}
- user:claim:{doi,openaire,orcid}

Structured Type: Qualifier

Property	Type	Multiplicity	Description
Class Id	String	1..1	
Class Name	String	1..1	
Scheme Id	String	1..1	
Scheme Name	String	1..1	

Structured Type: Instance

Results are always associated to one or more instances of the results, in the sense that different "manifestations" of the same result may exist and be comprised in the same entity. For example, the same article may be kept in two different repositories, both exposing the payload file (e.g., PDF) at different internet locations (URLs). Moreover, an instance of a Result is represented as a combination of one or more web resources, i.e. URLs to the files of the Result (e.g. article PDFs, dataset files), and are associated to the data sources hosting such resources (e.g., repositories).

Hence each results points to a list of instances, which are "surrogates" of the equivalent results it represents. If a result is not the result of a merge of equivalent results, it still has one instance (de facto replicating its properties).

Property	Type	Multiplicity	Description
Title	Structured(String,Qualifier,Provenance)	1..N	the titles of the Result, each with a typology represented by a Qualifier and Provenance information. Qualifier.scheme is dnet:dataCite_title.
Date of acceptance	Date	1..1	

Publisher	Structured(String,-,Provenance)	0..1	
Description	Structured(String,-,Provenance)	0..N	Contains the Abstract of the Result
PID	Structured(String,Qualifier,Provenance)	0..N	unique and persistent identifier used to identify the result together with the relative identification agency. The identification agency is expressed in the Qualifier part. Qualifier.scheme is dnet:pid_types
Access Rights	Qualifier	0..1	Maps dc:right, describes the access rights of the web resources relative to this instance. For Literature and Dataset the vocabulary is OPEN > EMBARGO > RESTRICTED > CLOSED > UNKNOWN
Embargo end-date	Date	0..1	Date when the embargo ends; empty if the licenseClass does not imply an embargo
Type	Qualifier	0..1	Type of the result instance. Values must comply to controlled vocabularies based on the specific subclass. For example type of publication instances must comply to the vocabulary dnet_publication_resource, while dataset instances to dnet:dataCite_resource. See also the OpenAIRE guidelines for dc:type and datacite:resourceType. In the case of software and other research product, the value of this property is chosen from a controlled vocabulary shaped by the community.
WebResource	String	1..N	URL of the files relative to the instance
Format	Structured(String,-,Provenance)	0..1	Maps the dc:format element, describes the file format of the web resources relative to this instance. Guidelines suggest the use of mime-types.

Structured Type: Structured

A Structured Property may contain a value whose type is injected by its instantiation (e.g. Structured(String)) and may require the presence of a Qualifier and/or Provenance information. In order to instantiate value, the types Qualifier and Provenance can be instantiated as exemplified below:

- Structured(String,Qualifier,Provenance): the type includes a value of type String, which can be qualified by given Scheme and Class, and also contains provenance information relative to the value (which process and/or which data source has provided the value);
- Structured(String,-,Provenance): the type includes a value of type String, without a qualifier, and also contains provenance information relative to the value (which process and/or which data source has provided the value);
- Structured(-, Qualifier, -): the type does not include a value and relative provenance information, but only a given qualifier Scheme and Class; in other words, it defines the usage of a controlled vocabulary.

Property	**Type**	**Multiplicity**	**Description**
Value	inherited from type instantiation	0..1	
Qualifier	Qualifier	0..1	

Provenance	Provenance	0..1	

Future extensions to the data model

The following entities are under definition and will be included in the model in the future.

Core entity: Person

A Person is here intended as the (metadata) description/representation of a person involved in the scholarly communication chain, such as scientific publications' authors, contributors, data scientists and project coordinators.

Property	Type	Multiplicity	Description
First name	Structured(String,-,Provenance)	0..1	
Family name	Structured(String,-,Provenance)	0..1	
Full name	Structured(String,-,Provenance)	0..1	
Fax	Structured(String,-,Provenance)	0..1	
Email	Structured(String,-,Provenance)	0..1	
Phone	Structured(String,-,Provenance)	0..1	
Nationality	Structured(-, Qualifier, -)	0..1	Nationality of the person. Qualifier.scheme is dnet:countries and its classes the standard ISO 3166-1 alpha-2 country codes
PID	Structured(String,Qualifier,Provenance)	0..N	unique and persistent identifier used to identify the person together with the relative identification agency. The identification agency is expressed in the Qualifier part. Qualifier.scheme is dnet:pid_types

Relationship	Target	Multiplicity	Description
CollectedFrom	Datasource	1..N	A Datasource from which this has been collected (e.g. an aggregator of institutional repositories, an entity registry)
Person_Project	Project	0..N	The Project this Person is somehow involved
Person_Result	Result	0..N	Result related (e.g. authored by) to this Person
Person_Person	Person	0..N	Link to other Person entities (e.g. co-authors)

Linking entity: Result_Person

Property	Type	Multiplicity	Description
Provenance	Provenance	1..1	
Qualifier	Qualifier	1..1	

Currently, only one Qualifier.Scheme is available for Result_Person relationship named "dnet:person_result_typologies", whose values are:

- hasAuthor: the result has been authored by the person
- isAuthorOf: inverse relationship of "hasAuthor"

Linking entity: Person_Person

Property	Type	Multiplicity	Description
Provenance	Provenance	1..1	
Qualifier	Qualifier	1..1	

Currently, only one Qualifier.Scheme is available for Person_Person relationship named "dnet:person_person_typologies", whose values are:

- isCoAuthorOf: relationship between two persons that co-authored a Publication. Currently not used by OpenAIRE.
- merges: A Person p1 merges p2 when the OpenAIRE de-duplication system identified a group of duplicate records (that includes p2) and generated a representative record p1 that merges them all.
- isMergedIn: inverse relationship of "merges"

Linking entity: Person_Project

Property	Type	Multiplicity	Description
Provenance	Provenance	1..1	
Qualifier	Qualifier	1..1	

Currently, only one Qualifier.Scheme is available for Result_Project relationship named "dnet:project_person_typologies", whose values are:

- isContact: the Person is the contact person for the project
- hasContact: inverse relationship of "isContact"