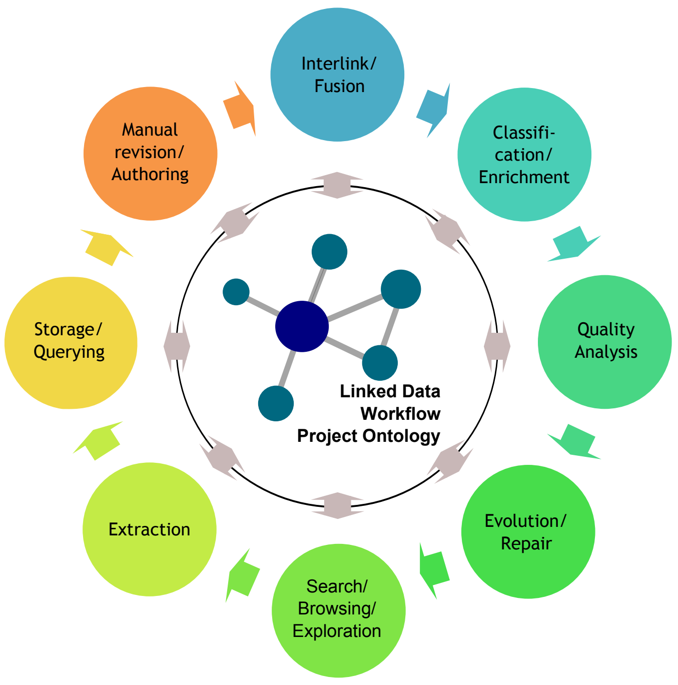

Interlink/ Fusion
Classifi- cation/ Enrichment
Quality Analysis
Evolution/ Repair
Search/ Browsing/ Exploration
Extraction
Storage/ Querying
Manual revision/ Authoring
Linked Data Workflow Project Ontology

# Linked Data Workflow Project Ontology

- LDWPO v. 1.0 -

Ontology Development Process

Technical Report

(Document Version 0.1 – )

(under construction)

TEAM

Sandro Rautenberg

Ivan Ermilov

Edgard Marx

Sören Auer, Prof. Dr.

Axel-C. Ngonga. Ngomo, Dr.

April 29, 2015

srautenberg@unicentro.br

# Contents

# 1 Introduction

This report presents the ontology development process applied to establish the *Linked Data Workflow Project ontology* (LDWPO). This process is the result of the combination of some methodological artifacts and best practices from On-to-Knowledge [6], METHONTOLOGY [3], and Ontology Development 101 Guide [4]. In a nutshell, this combination is based on:

- **On-to-Knowledge** contributes to the specification activity, by applying competence questions to confirm the purpose and scope of an ontology;
- **Ontology Development 101** defines an interactive process in an ontology development; and
- **METHONTOLOGY** suggests a set of artifacts to document and evaluate the ontology development process.

Considering the listed methodologies, we define and use a process based on four main activities: i) *ontology specification*; ii) *knowledge acquisition*; iii) *ontology implementation*; iv) and *ontology verification*. The Figure 1 represents this process, relating all inner tasks for each activity.

In the next sections, we discuss briefly each of those activities, presenting the artifacts used for documenting the LDWPO development.

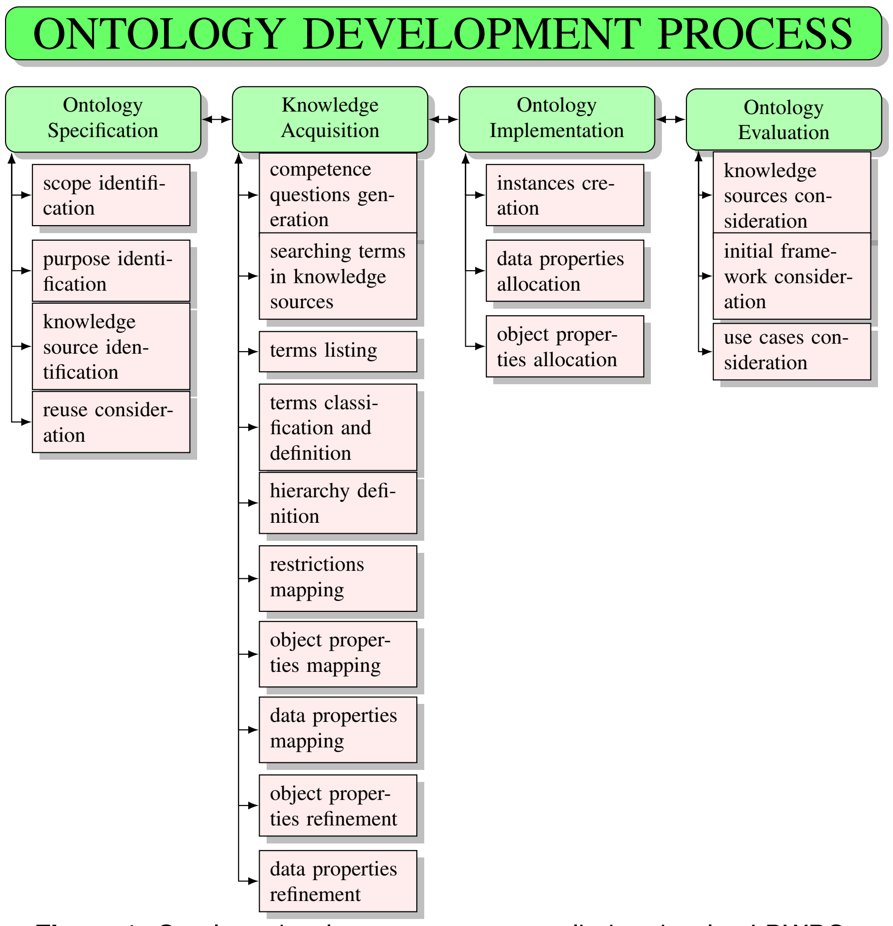

ONTOLOGY DEVELOPMENT PROCESS
Ontology Specification
scope identification
purpose identification
knowledge source identification
reuse consideration
Knowledge Acquisition
competence questions generation
searching terms in knowledge sources
terms listing
terms classification and definition
hierarchy definition
restrictions mapping
object properties mapping
data properties mapping
object properties refinement
data properties refinement
Ontology Implementation
instances creation
data properties allocation
object properties allocation
Ontology Evaluation
knowledge sources consideration
initial framework consideration
use cases consideration

**Figure 1:** Ontology development process applied to develop LDWPO.

# 2 Ontology Specification

This activity defines ontology development costs, taking into account:

- **The purpose identification** underlies why ontology has to be developed in regard to the existing ontology space.
- **The scope identification** answers the general questions as “who are the users?”, “what are the intention for using it?”, and so on.
- **The knowledge source identification** enumerates books, papers, dictionaries, and other sources with relevant information.
- **The reuse consideration** selects existing ontologies and vocabularies as candidates to provide established pieces of knowledge.

## 2.1 Identifying the purpose

The purpose of LDWPO addresses some provenance and reproducibility issues in a *Linked Data Workflow Project* (LDWProject). It models simple piece of knowledge about *project*s, *method*s, *plan*s, *execution*s, *artifact*s, and *people*, to describe the producing of RDF datasets in a systematic way. In an management point of view, this supports the maintainability of RDF dataset over time, considering the *Linked Data Lifecycle* [2].

## 2.2 Identifying the scope

1. Managing the lifecycle of RDF dataset maintaining;
2. Generating useful documents for reproducing the RDF dataset maintaining workflows; and
3. Mediating the use of *Linked Data Stack* technologies [1] for (semi-)automatized execution of workflows.

## 2.3 Identifying the knowledge sources

1. ***Dublin Core* (DC)[1] -** is a simple and powerful vocabulary used to describe web resources, promoting interoperability in Linked Data environments.

2. ***Description of a Project* (DOAP)[2] -** is a project to create an XML/RDF vocabulary to describe software projects, and in particular open source. Its use could be justify by contributing with a common vocabulary for projects.

3. ***Friend of a Friend* (FOAF)[3] -** is a project devoted to linking people and information using the Web. Its definitions could be used to link people to its actions, according the roles like contributor or creator.

4. ***Open Provenance Model Vocabulary* (OPMV)[4] -** is a lightweight provenance vocabulary to assist the interoperability between provenance information on the Semantic Web. Using together with other vocabularies, such as DC and FOAF, it could provide resources for publishing.

5. ***The PROV Ontology* (PROV-O)[5] -** is used to represent and interchange provenance information generated in different systems and under different contexts. It could be used to share useful definitions.

6. ***Publishing Workflow Ontology* (PWO)[6] -** is an ontology for describing the workflow associated with the publication of a document. Also, it could be used to share useful definitions.

7. ***Agile Knowledge Engineering and Semantic Web Research Group* (AKSW)[7] -** is a group compromised to research/develop methods and tools for Linked Data in an engineering fashion. The site of this group can provide useful definitions to ontology resources, specially, to create the ontology instances.

8. ***DBpedia*[8] -** is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web. Its knowledge base can be used for searching some ontology definitions.

[1]`http://purl.org/dc/terms/`
[2]`https://github.com/edumbill/doap`
[3]`http://xmlns.com/foaf/0.1/`
[4]`http://purl.org/net/opmv/ns`
[5]`http://www.w3.org/ns/prov#`
[6]`http://purl.org/spar/pwo`
[7]`http://aksw.org/About.html`
[8]`http://dbpedia.org/About`

## 2.4 Considering the reuse

We consider reusing resources from all the knowledge sources listed above.

# 3 Knowledge Acquisition

This activity comprehends the conceptualization and formalization stages in an ontology development process. Knowledge acquisition could be seen as the point of major interaction among the knowledge engineers and domain experts. Here, most of ontology's elements are listed and defined. Interactively, it considers the following tasks:

- **Competence questions generation** includes interviewing the domain experts, asking them to elaborate the questions that the ontology must answer. It is an easy way to get/understand the terms, relationships used in a particular domain.
- **The terms listing** takes into account the competence questions, enumerating the common vocabulary used by the domain experts. Commonly, two lists are formed: *substantives*, for identify classes and data properties; and *verbs* for discovering inner object properties.
- **Searching terms in knowledge sources** considers the knowledge sources, enumerating common terms from the domain that did not appear in the set of competence questions.
- **The terms classification and definition** arranges the consensual knowledge of the domain, classifying the terms as classes, object properties, data properties, instances, or restrictions. In addition, it is needed to define the terms in natural language, resulting on an established consensus about the meanings.
- **The hierarchy definition** organizes the classes as a tree once the terms are classified (in a similar fashion as in Object Oriented Programming). Also takes care on inheritance of the data and object properties.
- **The restrictions mapping** verifies the rules that can restrict the content of data or object properties for each class.
- **The object properties mapping** assembles, for each class, the terms considered object properties that explicitly show the relationship from a particular class to another class(es) of the domain.
- **The data properties mapping** aggregates, for each class, the terms considered data properties that explicitly are features of the class.
- **The object properties refinement** sets up the functionality, transitivity, symmetrical, and reflexive features for each object property.
- **The data properties refinement** defines the supported datatype and its functionality feature for each data property.

## 3.1 Generating the competence questions

Below, we list the competence questions organized according the Zachmann Framework [5].

### 3.1.1 Dimension What

1. What are the people's names that contribute in this project?
2. What is the person's name that create this project?
3. What is the project's name?
4. What were the employed tools in this project?
5. What are the input datasets of this project?
6. What are the datasets of this project?
7. What are the stages of this project?
8. What are the transformations in this project?
9. What is the output dataset of this stage?
10. What steps are done in this stage?
11. What is the previous step of this step?
12. What is the next step of this step?
13. What is the format of the dataset?
14. What are the project's goals?
15. What is the license of the dataset?
16. What kind of task is this step?
17. What tasks do this tool be applied?

### 3.1.2 Dimension Where

18. Where is the input dataset stored?
19. Where is the output dataset stored?

### 3.1.3 Dimension When

20. When was the step executed?
21. When did the workflow start?
22. When did the workflow finish?

### 3.1.4 Dimension Why

- *No questions were made.*

### 3.1.5 Dimension Who

23. Who are the people that contribute in this project?
24. Who is responsible for this step?

### 3.1.6 Dimension How

25. How is the input dataset stored?
26. How is the output dataset stored?

## 3.2 Listing terms from competence questions

- **Substantives** - dataset, format, input_dataset, license, next_step, output_dataset, person (people), person_name, previous_step, project, project_name, project_goal, responsible (contributor), stage, step, task, tool (employed tool), transformation, and workflow.
- **Verbs (in infinitive form)** - apply, be, contribute, create, do, employ, execute, finish, start, and store.

## 3.3 Enumerating the potential resources coming from knowledge sources

| Knowledge source | Terms |
|---|---|
| **DC** | **Classes -** Agent, AgentClass, Dataset, Event, ProvenanceStatement, and Software. **Properties -** abstract, contributor, created, creator, date, description, format, hasVersion, identifier, provenance, publisher, rights, and title. |
| **DOAP** | **Classes -** Project, Version, Specification, and Repository. **Properties -** name, homepage, created, shortdesc, description, maintainer, developer, documenter, tester, helper, and audience. |
| **FOAF** | **Classes -** Agent, Document, Group, Organization, Person, and Project. **Properties -** homepage, maker, mbox, and name. |
| **OPMV** | **Classes -** Agent, Artifact, and Process. **Properties -** used, wasControlledBy, wasDerivedFrom, wasEncodedBy, wasGeneratedBy, wasPerformedBy, wasTriggeredBy, and wasUsedAt. |
| **PROV-O** | **Classes -** Activity, Agent, Creator, Derivation, Organization, Person, Plan/plan, Publisher, and SoftwareAgent. |
| **PWO** | **Classes -** Step, and Workflow. **Properties -** hasFirstStep, hasNextStep, hasPreviousStep, hasStep, involvesEvent, isInvolvedInStep, isNeededBy, isProducedBy, isStepOf, needs, and produces. |
| **AKSW and/or LINKED DATA STACK** | **Stage's instances -** Extraction, Storage/Querying, Manual Revision/Authoring, Interlink/Fusing, Classification/Enrichment, Quality analysis, Evolution/Repair, and Searching/Browsing/Exploration. **Tool's instances -** Virtuoso Sponger, DBpedia Spotlight, DBpedia Spotlight UI, pool party extractor, D2R, R2R, Apache Stanbol, CSVImport, Virtuoso 7 RDF Store, SparQLed, sparqlify, SIREn, OntoWiki, RDFAuthor, poolparty, LIMES, Silk, Sieve, DL-Learner, ORE, LODrefine, Sigma, Spatial Semantic Browser, CubeViz, and Facete. |

**Table 1:** List of potential terms obtained from another knowledge sources.

## 3.4 Defining classes, data and object properties

### 3.4.1 Defining classes

1. **Artifact**. Abstract class that provides the `name` to the elements used to establish an `LDWStep`.
   *rdf:label - Artifact.*
   *owl:equivalentClass - [opmv:Artifact, and prov:Entity].*

2. **Dataset**. Subclass of `Artifact` that represents a collection of data. Related to a `LDWStep`, it is used as an input or an output `Artifact`.
   *rdf:label - Dataset.*
   *owl:equivalentClass - [dc:Dataset].*

3. **RDFDataset**. Subclass of `Dataset` that represents a collection of data adherent to the RDF data model, consisting of triples with subject, predicate and object.
   *rdf:label - RDFDataset.*

4. **Homepage**. Subclass of `Artifact` that represents the the initial or main web page of an `LDWProject`.
   *rdf:label - Homepage.*

5. **Report**. Subclass of `Artifact` that represents a document generated by aggregating the `Messages` and `Statuses` of an `LDWorkflowExecution`.
   *rdf:label - Report.*

6. **Tool**. Subclass of `Artifact` that represents a computational implementation of one or more algorithm(s) suitable for processing data.
   *rdf:label - Tool.*
   *owl:equivalentClass - [dc:Software, prov:SoftwareAgent, foaf:Agent].*

7. **ToolConfiguration**. Subclass of `Artifact` that represents a the setup file to be used by a `Tool`, when defining a `LDWStep`.
   *rdf:label - ToolConfiguration.*

8. **Format**. Class that represents the encode format of a `Dataset`, such as SQL, CSV, or RDF.
   *rdf:label - Format.*
   *owl:equivalentClass - [dc:MediaTypeOrExtent].*

9. **KnowledgeBody**. Top abstract class for representing pieces of knowledge used on modeling an `LDWProject`, providing the `name` and `description` data properties.
   *rdf:label - KnowledgeBody.*

10. **Condition**. Subclass of `KnowledgeBody` that represents (i) the obligations to be cover before an **LDWorkflowExecution** is performed (object property: precondition), and (ii) the settings achieved after the **LDWorkflowExecution** is successfully finished (object property: postcondition).
    *rdf:label - Condition.*

11. **Execution**. Subclass of `KnowledgeBody` and abstract class for `LDWorkflowExecution` and `LDWStepExecution` used for describing the execution issues an `LDWProject`. It describes a particular point of time with `startedDate` and `endedDate` data properties.
*rdf:label - Execution.*

12. **LDWorkflowExecution**. Subclass of `Execution` related to a particular execution of an `LDWorkflow`. Inside, it organizes a linked list of `LDWStepExecutions`.
*rdf:label - LDWorkflowExecution.*

13. **LDWStepExecution**. Subclass of `Execution` and `Step` that represents an atomic unit of `LDWorkflowExecution`. It is related to a particular `LDWStep`.
*rdf:label - LDWStepExecution.*

14. **LDWProject**. Subclass of `KnowledgeBody` that represents an endeavour of creating or maintaining `RDFDatasets`.
*rdf:label - LDWProject.*
*owl:equivalentClass - [foaf:Project].*

15. **Method**. Subclass of `KnowledgeBody` and abstract class for `Process`, `Stage`, and `Task` used for describing the methodological issues an `LDWProject`.
*rdf:label - Method.*

16. **Process**. Subclass of `Method` that represents a concise set of `Stages` used to maintain an `RDFDataset`.
*rdf:label - Process.*

17. **Stage**. Subclass of `Method` that represents a concise set of `Tasks` used to maintain an `RDFDataset`.
*rdf:label - Stage.*

18. **Task**. Subclass of `Method` that represents the atomic unit of an established `Process`. It describes a best-practice for maintaining an `RDFDataset`.
*rdf:label - Task.*

19. **Plan**. Subclass of `KnowledgeBody` and abstract class for `LDWorkflow` and `LDWStep` used for describing the planning issues an `LDWProject`.
*rdf:label - Plan.*

20. **LDWorkflow**. Subclass of `LDWorkflow` related to describe a particular planning activity of an `LDWProject` for maintaining an `RDFDataset`. Inside, it organizes a linked list of `LDWSteps`.
*rdf:label - LDWorkflow.*
*owl:equivalentClass - [pwo:Workflow].*

21. **LDWStep**. Subclass of `Plan` and `Step` that represents an atomic unit of `LDWorkflow`.
*rdf:label - LDWStep.*
*owl:equivalentClass - [pwo:Step, opmv:Process, and prov:Activity].*

22. **License**. Class that represents the type of access permission to a `Dataset`.
    *rdf:label - License.*
    *owl:equivalentClass - [dc:LicenseDocument].*

23. **Location**. Class that represents the path to access an `Artifact`.
    *rdf:label - Location.*
    *owl:equivalentClass - [prov:Location].*

24. **Message**. Class that represents the feedback report originating from applying a `Tool` and a `ToolConfiguration` in order to modify an `Dataset` during an `LDWStepExecution`.
    *rdf:label - Message.*

25. **Person**. Class that represents people involved in an `LDWProject`, performing the roles of contributor (for `Steps`) or creator (for whole `LDWProject`).
    *rdf:label - Person.*
    *owl:equivalentClass - [opmv:Agent, prov:Agent, prov:Person, dc:Agent, foaf:Agent, and foaf:Person].*

26. **Status**. Class that represents the classification for a `Message` (successful finished, unsuccessful finished, aborted, etc.).
    *rdf:label - Status.*

27. **Step**. Abstract class for `LDWStep` and `LDWStepExecution` used to describe the previous and next step for a given `Step`. It is used to define the linked list of the `LDWorkflow` and `LDWorkflowExecution`. It is related to a particular contributor (`Person`).
    *rdf:label - Step.*

### 3.4.2 Defining object properties

1. **contributor**. Object property that points to a `Person` who performed a `Step`. As restrictions, one `Step` can be performed by one `Person` and one *Person* can be related to one or more `Steps`.
   *rdf:label - contributor.*
   *owl:equivalentProperty - [opmv:wasControlledBy, opmv:wasPerformedBy, prov:wasAssociatedWi*
   *dc:contributor].*

2. **creator**. Object property that points to a `Person` who created an `LDWProject`. As restrictions, one `LDWProject` is created by one or more `Person` and one `Person` can be related to one or more `LDWProjects`.
   *rdf:label - creator.*
   *owl:equivalentProperty - [opmv:wasControlledBy, and dc:creator].*

3. **firstLdwStep**. Object property that points to initial `LDWStep` of an `LDWorkflow`. As restrictions, one `LDWorkflow` has one `LDWStep` as the `firstLdwStep` and one `LDWStep` can be the `firstLdwStep` in one or more `LDWorkflows`.

*rdf:label - firstLdwStep.*
*owl:equivalentProperty - [pwo:hasFirstStep].*

4. **firstLdwStepExecution** Object property that points to initial `LDWStepExecution` of an `LDWorkflowExecution`. As restrictions, one `LDWorkflowExecution` has one `LDWStepExecution` as `firstLdwStepExecution` and one `LDWStepExecutio` is the `firstLdwStepExecution` of a single `LDWorkflowExecution`.
   *rdf:label - firstLdwStepExecution.*

5. **format**. Object property that points to the `Format` of a `Dataset`. As restrictions, one `Format` is related to one or more `Datasets` and a `Dataset` is related to a single `Format`.
   *rdf:label - format.*
   *owl:equivalentProperty - [dc:format].*

6. **homepage**. Object property that points to the `Homepage` of a `LDWProject`. As restrictions, one `Homepage` is related to one or more `LDWProjects` and a `LDWProject` is related to a single `Homepage`.
   *rdf:label - homepage.*
   *owl:equivalentProperty - [foaf:homepage].*

7. **inputDataset**. Object property that points to the `Datasets` used as input by an `LDWStep`. As restrictions, one `LDWStep` has one or more `Dataset` as input and one `Dataset` can be related as input to one or more `LDWSteps`.
   *rdf:label - inputDataset.*
   *owl:equivalentProperty - [pwo:needs, opmv:used, and prov:used].*

8. **ldWorkflow**. Object property that points to the `LDWorkflows` related to an `LDWProject`. As restrictions, one `LDWProject` has one or more `LDWorkflows` and one `LDWorkflow` is related to single `LDWorkflow`.
   *rdf:label - ldWorkflow.*

9. **ldWorkflowExecution**. Object property that points to the `LDWorkflowExecutions` related to an `LDWorklow`. As restrictions, one `LDWorkflow` has one or more `LDWorkflowExecutions` and one `LDWorkflowExecution` is related to single `LDWorkflow`.
   *rdf:label - ldWorkflowExecution.*

10. **ldwStep**. Object property that points to the `LDWSteps` related to an `LDWorklow`. As restrictions, one `LDWorkflow` has one or more `LDWSteps` and one `LDWStep` is related to one or more `LDWorkflow`.
    *rdf:label - ldwStep.*
    *owl:equivalentProperty - [pwo:hasStep].*

11. **ldwStepExecution**. Object property that points to the `LDWStepExecutions` related to an `LDWorklowExecution` and an `LDWStep`. As restrictions, one `LDWStepExecution` is related to a single `LDWStep` and to a single `LDWorkflowExecutior` and both `LDWorkflowExecution` and `LDWStep` are related to one or more

`LDWStepExecution`.
*rdf:label - ldwStepExecution.*

12. **license**. Object property that points to the type of `License` of a `Dataset`. As restrictions, one `Dataset` has one `License` and one `License` is related to one or more `Datasets`.
*rdf:label - license.*
*owl:equivalentProperty - [dc:license].*

13. **location**. Object property that points to the physical location of an `Artifact`, which could be an URL or local filesystem path. As restrictions, one `Artifact` has one `Location` and one `Location` is related to one or more `Artifacts`.
*rdf:label - location.*
*owl:equivalentProperty - [prov:atLocation].*

14. **message**. Object property that points to the `Message` generated as feedback when a `LDWStepExecution` is executed.
As restrictions, one `LDWStepExecution` has one `Message` and one `Message` is related to one `LDWStepExecution`.
*rdf:label - message.*

15. **nextStep**. Object property that points to the next `Step` into a linked list of `Steps`. It has `previousStep` as an inverse property. As restriction, one `Step` has only one next `Step` and is an irreflexive and asymmetric object property.
*rdf:label - nextStep.*
*owl:equivalentProperty - [pwo:hasNextStep].*

16. **outputDataset**. Object property that points to the `Datasets` to be transformed in an `LDWStep`. As restrictions, one `LDWStep` has one or more `Dataset` as output and one `Dataset` can be related as output to one or more `LDWSteps`.
*rdf:label - outputDataset.*
*owl:equivalentProperty - [pwo:produces, and prov:generated].*

17. **postcondition**. Object property that points to the `Condition` to achieve after an `LDWorkflow` is executed by an `LDWorkflowExecution`.
As restrictions, one `Workflow` has one or more `Conditions` as postcondition and one `Condition` can be related as postcondiction to one or more `LDWorkflows`.
*rdf:label - postcondition.*

18. **precondiction**. Object property that points to the `Condition` to be presented before an `LDWorkflow` is executed by an `LDWorkflowExecution`.
As restrictions, one `Workflow` has one or more `Conditions` as precondition and one `Condition` can be related as precondiction to one or more `LDWorkflows`.
*rdf:label - precondition.*

19. **previousStep**. Object property that points to the previous `Step` into a linked list of `Steps`. It has `nextStep` as an inverse property. As restriction, one `Step` has only one previous `Step` and is an irreflexive and asymmetric object property.

*rdf:label - previousStep.*
*owl:equivalentProperty - [pwo:hasPreviousStep, opmv:wasTriggeredBy, and prov:wasInformedBy*

20. **report**. Object property that points to the `Report` generated as feedback when a `LDWorkflowExecution` is executed. It groups all `Message`s and `Status`es from the related `LDWStepExecution`.
    As restriction, one `LDWorkflowExecution` has only one `Report` and one `Report` is related to only one `LDWorkflowExecution`.
    *rdf:label - report.*

21. **stage**. object property that points to the `Stage`s of a `Process`. As restrictions, one `Process` has one or more *Stage*s and one `Stage` is related to one or more `Process`es.
    *rdf:label - stage.*

22. **status**. Object property that points to the `Status` of executing a `LDWStepExecution`. As restriction, one `LDWStepExecution` has only one `Status` and one `Status` is related to one or more `LDWStepExecution`.
    *rdf:label - status.*

23. **task**. Object property that points to the `Task` of an `LDWStep` or `Stage`. As restrictions, one `Task` is related one or more *Stage*s and or or more `LDWSteps` and one `LDWStep` and one `Stage` are related to one or more `Task`s.
    *rdf:label - task.*

24. **tool**. Object property that points to the `Tool` employed in a `LDWStep`. As restrictions, one `Step` can employ one `Tool` and one `Tool` is related to one or more `LDWStep`s.
    *rdf:label - tool. owl:equivalentProperty - [pwo:needs, opmv:used, and prov:used].*

25. **toolConfiguration**. Object property that points to the `ToolConfiguration` used by a `Tool` in a `LDWStep`. As restrictions, one `Tool` uses one or more `ToolConfigurations` and one `LDWStep` uses one `ToolConfigurations`. And one `ToolConfiguration` can be used by one `Tool` and also used by one or more `LDWStep`.
    *rdf:label - toolConfiguration. owl:equivalentProperty - [pwo:needs, and opmv:used].*

### 3.4.3 Defining data properties

1. **command**. Data property that contains the console command of a planned `LDWStep`. It relates the `Tool` and `ToolConfiguration` to be used by an `LDWStepExecution`. As restriction, `command` is a functional property and accepts only xsd:String values.
   *rdf:label - command.*

2. **description**. Data property used by a `KnwoledgeBody` to describe instances. As restriction, `description` is a functional property and accepts only xsd:String values.

*rdf:label - description.*
*owl:equivalentProperty - [dc:description].*

3. **endedDate**. Data property that expresses when an `Execution` instance is finished. As restriction, `endedDate` is a functional property and accepts only xsd:DateTime values.
   *rdf:label - endedDate.*
   *owl:equivalentProperty - [dc:date, opmv:wasEndedAt, and prov:endedAtTime].*

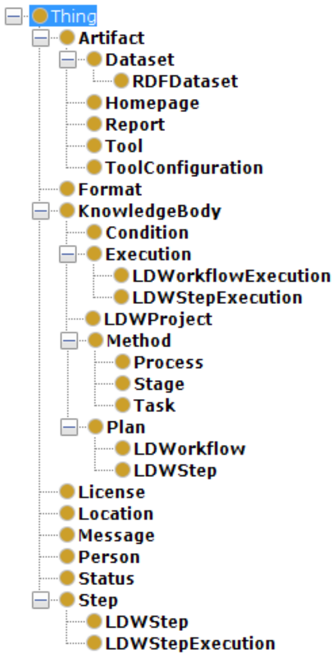
4. **goal**. Data property that expresses the set of goals of an `LDWProject`. As restriction, `goal` is a functional property and accepts only xsd:String values.
   *rdf:label - goal.*

5. **name**. Data property used by `Artifact`, `KnwoledgeBody`, `License`, and `Person`, to name instances. As restriction, `name` is a functional property and accepts only xsd:String values.
   *owl:equivalentProperty - [dc:identifier, foaf:name]. rdf:label - name.*

6. **startedDate.** Data property that expresses when an `Execution` instance is started. As restriction, `startedDate` is a functional property and accepts only xsd:DateTime values.
   *owl:equivalentProperty - [dc:date, dc:created, prov:startedAtTime, opmv:wasStartedAt].*
   *rdf:label - startedDate.*

7. **value**. Data property used by `Format`, `KnwoledgeBody`, `Location`, `Message`, and `Status`, to attribute a content to the instances. As restriction, `value` is a functional property and accepts only xsd:String format.
   *rdf:label - value.*

## 3.5 Defining class hierarchy



```
Thing
  Artifact
    Dataset
      RDFDataset
    Homepage
    Report
    Tool
    ToolConfiguration
  Format
  KnowledgeBody
    Condition
    Execution
      LDWorkflowExecution
      LDWStepExecution
    LDWProject
    Method
      Process
      Stage
      Task
    Plan
      LDWorkflow
      LDWStep
  License
  Location
  Message
  Person
  Status
  Step
    LDWStep
    LDWStepExecution
```

**Figure 2:** LDWPO class hierarchy.

## 3.6 Refining properties and mapping restrictions

Refining properties and mapping restrictions are interrelated tasks and can be performed at the same time.

To refine properties, it is needed to establish the domain and range of data and object properties.

To map restrictions, it is considered the characteristics of a particular property. For data properties, we establish the supported datatype and functionality feature. For object properties, we set up the functional, inverse functional, reflexive, irreflexive, symmetric, asymmetric, and transitive characteristics.

The results of refining and mapping tasks for LDWPO are expressed by tables, as shown below.

## 3.7 Mapping and refining object properties

| Object Property | Domain | Range | Characteristics |
|---|---|---|---|
| contributor | Step | Person | [F] |
| creator | LDWProject | Person | [–] |
| format | Dataset | Format | [F] |
| firstLdwStep | LDWorkflow | LDWStep | [F] |
| firstLdwStepExecution | LDWorkflowExecution | LDWStepExecution | [F, IF] |
| homepage | LDWProject | Homepage | [F] |
| inputDataset | LDWStep | Dataset | [–] |
| ldWorkflow | LDWProject | LDWorkflow | [IF] |
| ldWorkflowExecution | LDWorkflow | LDWorkflowExecution | [IF] |
| ldwStep | LDWorkflow | LDWStep | [–] |
| ldwStepExecution | LDWorkflowExecution, LDWStep | LDWStepExecution | [IF] |
| license | Dataset | License | [F] |
| location | Artifact | Location | [F] |
| message | LDWStepExecution | Message | [F, IF] |
| nextStep | Step | Step | [I, A] |
| outputDataset | LDWStep | Dataset | [IF] |
| postcondiction | LDWorkflow | Condition | [–] |
| precondiction | LDWorkflow | Condition | [–] |
| previousStep | Step | Step | [I, A] |
| report | LDWorkflowExecution | Report | [F, IF] |
| stage | Process | Stage | [–] |
| status | LDWorkflowExecution | Status | [F] |
| task | Stage, LDWStep | Task | [–] |
| tool | LDWStep | Tool | [F] |
| toolConfiguration | LDWStep, Tool | ToolConfiguration | [–] |

**Table 2:** Object properties - domains, characteristics, and ranges. For Characteristics: (F) - Functional, (IF) - Inverse Functional, (R) - Reflexive, (I) - Irreflexive, (S) - Symmetric, (A) - Asymmetric, and (T) - Transitive, (–) - no characteristic define

.

## 3.8 Mapping and refining data properties

| Data Property | Domain | Range | Functional |
|---|---|---|---|
| command | LDWStep | String | Yes |
| description | KnowledgeBody | String | Yes |
| endedDate | Execution | DateTime | Yes |
| goal | LDWProject | String | Yes |
| name | KnowledgeBody, Artifact, License, Person | String | Yes |
| startedDate | Execution | DateTime | Yes |
| value | Message, Status, Format, Location | String | Yes |

**Table 3:** Data properties - domains and ranges.

## 3.9 Ontology diagram

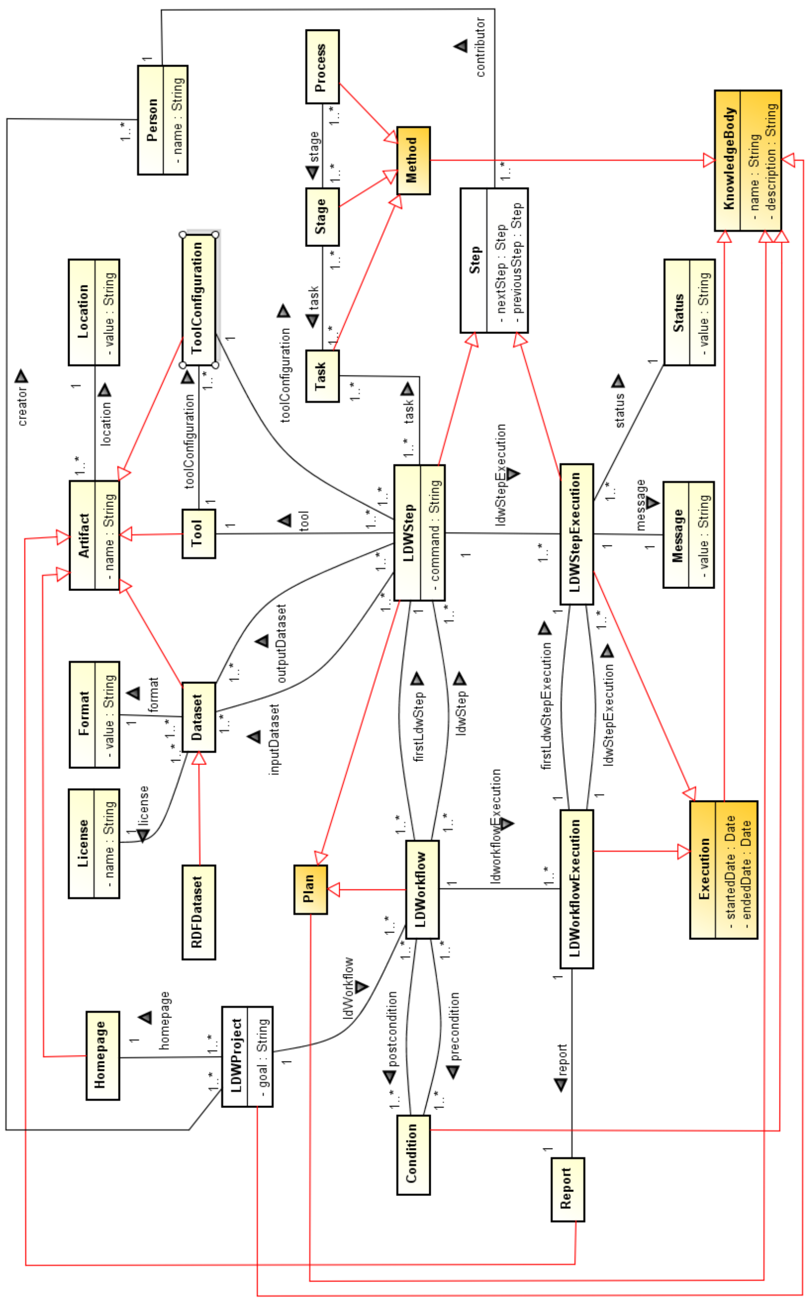


**Figure 3:** Representing LDWPO v. 1.0, highlighting the abstract concepts Knowledge Body, Method, Plan, and Execution.

# 4 Ontology Implementation

With minor interaction with domain experts, this activity is reserved to:

- **The instances creation** specifies the terms considered as samples of each class.
- **The data properties allocation** assigns the value(s) of internal properties for each instance.
- **The object properties allocation** connects a particular instance to the other instances of the domain.

As main result of this activity, the first version of LDWPO is available at `https://github.com/AKSW/ldwpo/blob/master/1.0/ldwpo.owl`.

# 5 Ontology Verification

Verification is related to check if an ontology meets the requirements by:

- **The knowledge sources consideration** compares the consistency of the represented knowledge in the ontology to the accepted understanding of the domain based on the knowledge sources. Basically, we relate the LDWPO resources to the resources of knowledge sources, according a common understanding about the inner definitions. The relations are expressed at the level of owl:equivalentClass and owl:equivalentProperty.

- **The initial framework consideration** confirms the accuracy and completeness of the ontology, taking into account the purpose, scope, and mainly, by prototyping the competence questions.

- **Use cases consideration** inspects the usefulness of the ontology by simulating some scenarios.

## 5.1 Knowledge source consideration

| Class | Equivalent class |
|---|---|
| Artifact | opmv:Artifact |
| Artifact | prov:Entity |
| Dataset | dc:Dataset |
| Tool | dc:Software |
| Tool | prov:SoftwareAgent |
| Tool | foaf:Agent |
| Format | dc:MediaTypeOrExtent |
| LDWProject | foaf:Project |
| LDWorkflow | pwo:Workflow |
| LDWStep | pwo:Step |
| LDWStep | opmv:Process |
| LDWStep | prov:Activity |
| License | dc:LicenseDocument |
| Location | prov:Location |
| Person | opmv:Agent |
| Person | prov:Agent |
| Person | prov:Person |
| Person | dc:Agent |
| Person | foaf:Agent |
| Person | foaf:Person |

**Table 4:** List of classes and its equivalence to another knowledge sources [owl:equivalentClass].

### 5.1.1 Knowledge source consideration - Classes

### 5.1.2 Knowledge source consideration - Object properties

| textbfObject property | **Equivalent object property** |
|---|---|
| contributor | opmv:wasControlledBy |
| contributor | opmv:wasPerformedBy |
| contributor | prov:wasAssociatedWith |
| contributor | dc:contributor |
| creator | opmv:wasControlledBy |
| creator | dc:creator |
| format | dc:format |
| firstLdwStep | pwo:hasFirstStep |
| homepage | foaf:homepage |
| inputDataset | pwo:needs |
| inputDataset | opmv:used |
| inputDataset | prov:used |
| ldwStep | pwo:hasStep |
| license | dc:license |
| location | prov:atLocation |
| nextStep | pwo:hasNextStep |
| outputDataset | pwo:produces |
| outputDataset | prov:generated |
| previousStep | pwo:hasPreviousStep |
| previousStep | opmv:wasTriggeredBy |
| previousStep | prov:wasInformedBy |
| tool | pwo:needs |
| tool | opmv:used |
| tool | prov:used |
| toolConfiguration | pwo:needs |
| toolConfiguration | opmv:used |

**Table 5:** List of object properties and its equivalence to another knowledge sources [owl:equivalentProperty].

### 5.1.3 Knowledge source consideration - Data properties

| textbfData property | **Equivalent data property** |
|---|---|
| description | dc:description |
| endedDate | opmv:wasEndedAt |
| endedDate | prov:endedAtTime |
| endedDate | dc:date |
| name | dc:identifier |
| name | foaf:name |
| startedDate | opmv:wasStartedAt |
| startedDate | prov:startedAtTime |
| startedDate | dc:date |

**Table 6:** List of data properties and its equivalence to another knowledge sources [owl:equivalentProperty].

## 5.2 Initial framework consideration

### 5.2.1 What are the people's names that contribute in this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameContributor WHERE {
  ?project            rdf:type                   ldwpo:LDWProject   .
  ?project            ldwpo:name                 ?nameProject       .
  ?project            ldwpo:ldWorkflow           ?ldworkflow        .
  ?ldworkflow         ldwpo:firstLdwStep         ?step              .
  ?ldworkflow         ldwpo:ldWorkflowExecution  ?workflowExecution .
  ?step               (ldwpo:nextStep)*          ?linkedStep        .
  ?linkedStep         ldwpo:ldwStepExecution     ?stepExecution     .
  ?workflowExecution  ldwpo:ldwStepExecution     ?stepExecution     .
  ?stepExecution      ldwpo:contributor          ?contributor       .
  ?contributor        ldwpo:name                 ?nameContributor   .
} ORDER BY ?project
```

### 5.2.2 What is the person's name that create this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameCreator WHERE {
  ?project  rdf:type       ldwpo:LDWProject .
  ?project  ldwpo:name     ?nameProject     .
  ?project  ldwpo:creator  ?creator         .
  ?creator  ldwpo:name     ?nameCreator     .
} ORDER BY ?project
```

### 5.2.3 What is the project's name?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject WHERE {
  ?project  rdf:type    ldwpo:LDWProject .
  ?project  ldwpo:name  ?nameProject     .
} ORDER BY ?project
```

### 5.2.4 What were the employed tools in this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameTool WHERE {
  ?project     rdf:type            ldwpo:LDWProject .
  ?project     ldwpo:name          ?nameProject     .
  ?project     ldwpo:ldWorkflow    ?ldworkflow      .
  ?ldworkflow  ldwpo:firstLdwStep  ?step            .
  ?step        (ldwpo:nextStep)*   ?linkedStep      .
  ?linkedStep  ldwpo:tool          ?tool            .
  ?tool        ldwpo:name          ?nameTool        .
} ORDER BY ?project
```

### 5.2.5 What are the input datasets of this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameDataset WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldworkflow       .
  ?ldworkflow    ldwpo:firstLdwStep  ?step             .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:inputDataset  ?inputDataset     .
  ?inputDataset  ldwpo:name          ?nameDataset      .
} ORDER BY ?project
```

### 5.2.6 What are the datasets of this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameDataset WHERE {
  {
    SELECT ?nameProject ?nameDataset WHERE {
      ?project       rdf:type            ldwpo:LDWProject  .
      ?project       ldwpo:name          ?nameProject      .
      ?project       ldwpo:ldWorkflow    ?ldworkflow       .
      ?ldworkflow    ldwpo:firstLdwStep  ?step             .
      ?step          (ldwpo:nextStep)*   ?linkedStep       .
      ?linkedStep    ldwpo:inputDataset  ?inputDataset     .
      ?inputDataset  ldwpo:name          ?nameDataset      .
    }
  }
  UNION
  {
    SELECT ?nameProject ?nameDataset WHERE {
      ?project       rdf:type             ldwpo:LDWProject  .
      ?project       ldwpo:name           ?nameProject      .
      ?project       ldwpo:ldWorkflow     ?ldworkflow       .
      ?ldworkflow    ldwpo:firstLdwStep   ?step             .
      ?step          (ldwpo:nextStep)*    ?linkedStep       .
      ?linkedStep    ldwpo:outputDataset  ?outputDataset    .
      ?outputDataset  ldwpo:name          ?nameDataset      .
    }
  }
} ORDER BY ?project
```

### 5.2.7 What are the stages of this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameStage WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldworkflow       .
  ?ldworkflow    ldwpo:firstLdwStep  ?step             .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:task          ?task             .
  ?stage         rdf:type            ldwpo:Stage       .
  ?stage         ldwpo:task          ?task             .
  ?stage         ldwpo:name          ?nameStage        .
} ORDER BY ?project
```

### 5.2.8 What are the transformations in this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?description WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldworkflow       .
  ?ldworkflow    ldwpo:firstLdwStep  ?step             .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:description   ?description      .
  ?ldworkflow    ldwpo:ldwStep       ?linkedStep       .
}
```

### 5.2.9 What is the output dataset of this stage?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameDataset WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldworkflow       .
  ?ldworkflow    ldwpo:firstLdwStep  ?step             .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:inputDataset  ?inputDataset     .
  ?inputDataset  ldwpo:name          ?nameDataset      .
} ORDER BY ?project
```

### 5.2.10 What steps are done in this stage?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?ldWorkflow ?stage ?linkedStep WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?stage         rdf:type            ldwpo:Stage       .
  ?step          rdf:type            ldwpo:LDWStep     .
  ?ldWorkflow    rdf:type            ldwpo:LDWorkflow  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldWorkflow       .
  ?ldWorkflow    ldwpo:firstLdwStep  ?step             .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:task          ?task             .
  ?stage         ldwpo:task          ?task             .
} ORDER BY ?project ?stage
```

### 5.2.11 What is the previous step of this step?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?ldWorkflow ?linkedStep ?previousStep WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?step          rdf:type            ldwpo:LDWStep     .
  ?linkedStep    rdf:type            ldwpo:LDWStep     .
  ?previousStep  rdf:type            ldwpo:LDWStep     .
  ?ldWorkflow    rdf:type            ldwpo:LDWorkflow  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldWorkflow       .
  ?ldWorkflow    ldwpo:firstLdwStep  ?step             .
  ?ldWorkflow    ldwpo:ldwStep       ?linkedStep       .
  ?ldWorkflow    ldwpo:ldwStep       ?previousStep     .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:previousStep  ?previousStep     .
} ORDER BY ?project
```

### 5.2.12 What is the next step of this step?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?ldWorkflow ?linkedStep ?nextStep WHERE {
  ?project       rdf:type            ldwpo:LDWProject  .
  ?step          rdf:type            ldwpo:LDWStep     .
  ?linkedStep    rdf:type            ldwpo:LDWStep     .
  ?previousStep  rdf:type            ldwpo:LDWStep     .
  ?ldWorkflow    rdf:type            ldwpo:LDWorkflow  .
  ?project       ldwpo:name          ?nameProject      .
  ?project       ldwpo:ldWorkflow    ?ldWorkflow       .
  ?ldWorkflow    ldwpo:firstLdwStep  ?step             .
  ?ldWorkflow    ldwpo:ldwStep       ?linkedStep       .
  ?ldWorkflow    ldwpo:ldwStep       ?nextStep         .
  ?step          (ldwpo:nextStep)*   ?linkedStep       .
  ?linkedStep    ldwpo:nextStep      ?nextStep         .
} ORDER BY ?project
```

### 5.2.13 What is the format of the dataset?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameDataset ?format WHERE {
  {
    SELECT DISTINCT ?dataset ?format WHERE {
      ?dataset     rdf:type             ldwpo:Dataset    .
      ?step        rdf:type             ldwpo:LDWStep    .
      ?format      rdf:type             ldwpo:Format     .
      ?step        ldwpo:inputDataset   ?dataset         .
      ?dataset     ldwpo:format         ?format          .
    }
  }
  UNION
  {
    SELECT DISTINCT ?dataset ?format WHERE {
      ?dataset     rdf:type             ldwpo:Dataset    .
      ?step        rdf:type             ldwpo:LDWStep    .
      ?format      rdf:type             ldwpo:Format     .
      ?step        ldwpo:outputDataset  ?dataset         .
      ?dataset     ldwpo:format         ?format          .
    }
  }
} ORDER BY ?dataset
```

### 5.2.14 What are the project's goals?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?goal WHERE {
  ?project  rdf:type    ldwpo:LDWProject .
  ?project  ldwpo:name  ?nameProject     .
  ?project  ldwpo:goal  ?goal            .
}
```

### 5.2.15 What is the license of the dataset?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameDataset ?nameLicense WHERE {
  ?dataset  rdf:type       ldwpo:Dataset .
  ?license  rdf:type       ldwpo:License .
  ?dataset  ldwpo:name     ?nameDataset  .
  ?dataset  ldwpo:license  ?license      .
  ?license  ldwpo:name     ?nameLicense  .
}
```

### 5.2.16 What kind of task is this step?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?step ?task ?taskDescription WHERE {
  ?step  rdf:type           ldwpo:LDWStep     .
  ?task  rdf:type           ldwpo:Task        .
  ?step  ldwpo:task         ?task             .
  ?task  ldwpo:description  ?taskDescription  .
}
```

### 5.2.17 What tasks do this tool be applied?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?task ?tool WHERE {
  ?step  rdf:type    ldwpo:LDWStep  .
  ?task  rdf:type    ldwpo:Task     .
  ?tool  rdf:type    ldwpo:Tool     .
  ?step  ldwpo:task  ?task          .
  ?step  ldwpo:tool  ?tool          .
}
```

### 5.2.18 Where is the input dataset stored?

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?dataset ?location WHERE {
  ?dataset   rdf:type            ldwpo:Dataset  .
  ?step      rdf:type            ldwpo:LDWStep  .
  ?location  rdf:type            ldwpo:Location .
  ?step      ldwpo:inputDataset  ?dataset       .
  ?dataset   ldwpo:location      ?location      .
}
```

### 5.2.19 Where is the output dataset stored?

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?dataset ?location WHERE {
  ?dataset   rdf:type             ldwpo:Dataset  .
  ?step      rdf:type             ldwpo:LDWStep  .
  ?location  rdf:type             ldwpo:Location .
  ?step      ldwpo:outputDataset  ?dataset       .
  ?dataset   ldwpo:location       ?location      .
}
```

### 5.2.20 When was the step executed?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?stepExecution ?startedDate WHERE {
  ?stepExecution  rdf:type           ldwpo:LDWStepExecution  .
  ?stepExecution  ldwpo:startedDate  ?startedDate            .
}
```

### 5.2.21 When did the workflow start?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?workflowExecution ?startedDate WHERE {
  ?workflowExecution  rdf:type           ldwpo:LDWorkflowExecution  .
  ?workflowExecution  ldwpo:startedDate  ?startedDate               .
}
```

### 5.2.22 When did the workflow finish?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?workflowExecution ?endedDate WHERE {
  ?workflowExecution  rdf:type         ldwpo:LDWorkflowExecution  .
  ?workflowExecution  ldwpo:endedDate  ?endedDate                .
}
```

### 5.2.23 Who are the people that contribute in this project?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?nameProject ?nameContributor WHERE {
  ?project            rdf:type                   ldwpo:LDWProject   .
  ?project            ldwpo:name                 ?nameProject       .
  ?project            ldwpo:ldWorkflow           ?ldworkflow        .
  ?ldworkflow         ldwpo:firstLdwStep         ?step              .
  ?ldworkflow         ldwpo:ldWorkflowExecution  ?workflowExecution .
  ?step               (ldwpo:nextStep)*          ?linkedStep        .
  ?linkedStep         ldwpo:ldwStepExecution     ?stepExecution     .
  ?workflowExecution  ldwpo:ldwStepExecution     ?stepExecution     .
  ?stepExecution      ldwpo:contributor          ?contributor       .
  ?contributor        ldwpo:name                 ?nameContributor   .
} ORDER BY ?project
```

### 5.2.24 Who is responsible for this step?

```
PREFIX rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:   <http://www.w3.org/2002/07/owl#>
PREFIX xsd:   <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?stepExecution ?nameContributor WHERE {
  ?stepExecution  rdf:type           ldwpo:LDWStepExecution  .
  ?stepExecution  ldwpo:contributor  ?contributor            .
  ?contributor    ldwpo:name         ?nameContributor        .
}
```

### 5.2.25 How is the input dataset stored?

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?dataset ?format WHERE {
  ?dataset     rdf:type            ldwpo:Dataset    .
  ?step        rdf:type            ldwpo:LDWStep    .
  ?format      rdf:type            ldwpo:Format     .
  ?step        ldwpo:inputDataset  ?dataset         .
  ?dataset     ldwpo:format        ?format          .
}
```

### 5.2.26 How is the output dataset stored?

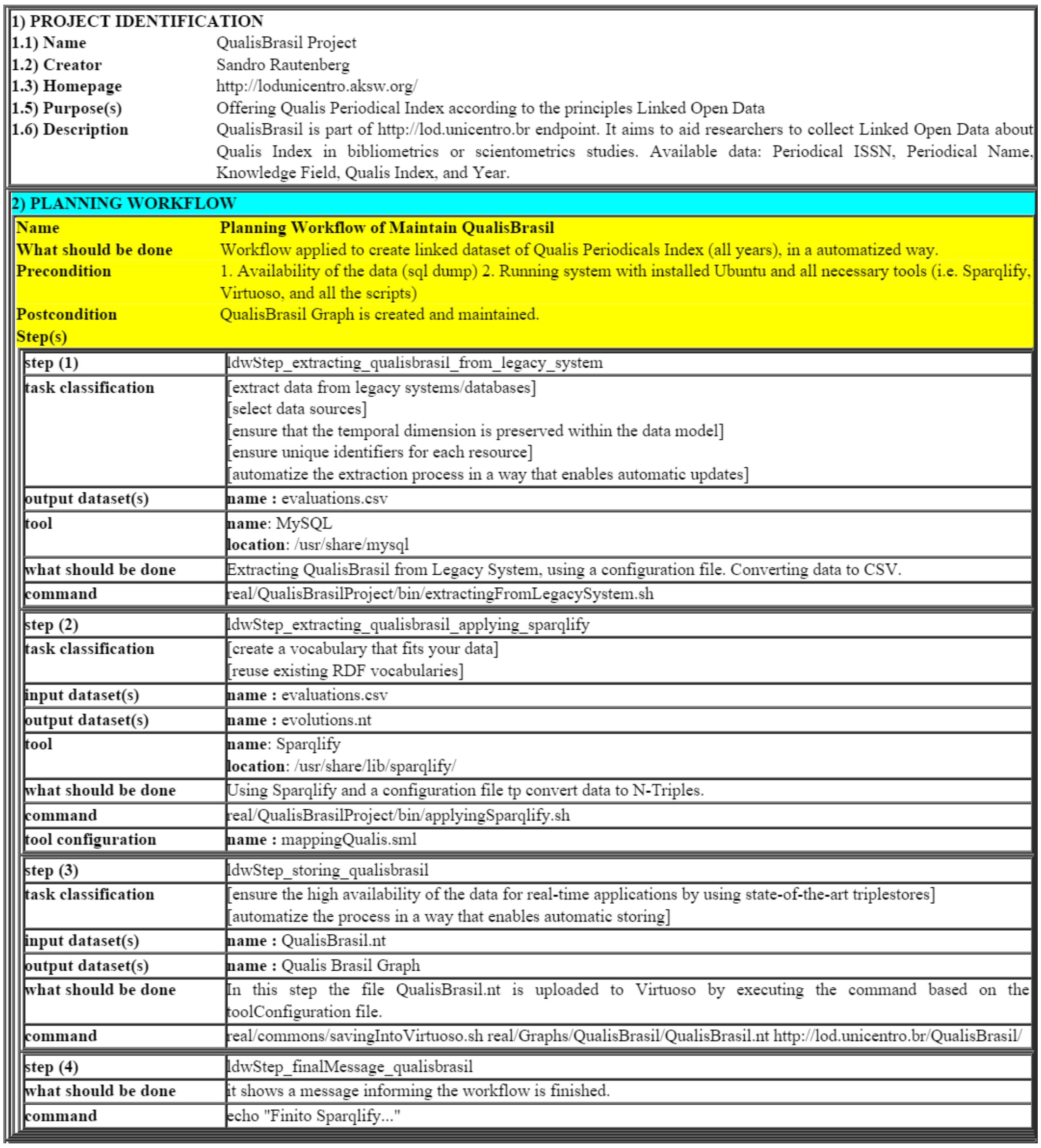
```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl:  <http://www.w3.org/2002/07/owl#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ldwpo: <http://ldwpo.aksw.org/terms/1.0/>

SELECT DISTINCT ?dataset ?format WHERE {
  ?dataset     rdf:type             ldwpo:Dataset    .
  ?step        rdf:type             ldwpo:LDWStep    .
  ?format      rdf:type             ldwpo:Format     .
  ?step        ldwpo:outputDataset  ?dataset         .
  ?dataset     ldwpo:format         ?format          .
}
```

## 5.3 Use cases consideration

Next, we presents qualisConversion and qualisInterlinking LDWProjects as real-world use cases of LDWPO. They are represented by partial reports generated by the reportgen tool (see subsection 6.1) and were executed in 25-04-2015, using the ldStackWorkflow tool (see subsection 6.2). As result of the executions, Qualis data is now available as Linked Open Data (see the VoID description listed in subsection 6.3).

### 5.3.1 qualisConversion LDWProject



| **1) PROJECT IDENTIFICATION** | |
|---|---|
| **1.1) Name** | QualisBrasil Project |
| **1.2) Creator** | Sandro Rautenberg |
| **1.3) Homepage** | http://lodunicentro.aksw.org/ |
| **1.5) Purpose(s)** | Offering Qualis Periodical Index according to the principles Linked Open Data |
| **1.6) Description** | QualisBrasil is part of http://lod.unicentro.br endpoint. It aims to aid researchers to collect Linked Open Data about Qualis Index in bibliometrics or scientometrics studies. Available data: Periodical ISSN, Periodical Name, Knowledge Field, Qualis Index, and Year. |

| **2) PLANNING WORKFLOW** | |
|---|---|
| **Name** | **Planning Workflow of Maintain QualisBrasil** |
| **What should be done** | Workflow applied to create linked dataset of Qualis Periodicals Index (all years), in a automatized way. |
| **Precondition** | 1. Availability of the data (sql dump) 2. Running system with installed Ubuntu and all necessary tools (i.e. Sparqlify, Virtuoso, and all the scripts) |
| **Postcondition** | QualisBrasil Graph is created and maintained. |
| **Step(s)** | |

| | |
|---|---|
| **step (1)** | ldwStep_extracting_qualisbrasil_from_legacy_system |
| **task classification** | [extract data from legacy systems/databases]<br>[select data sources]<br>[ensure that the temporal dimension is preserved within the data model]<br>[ensure unique identifiers for each resource]<br>[automatize the extraction process in a way that enables automatic updates] |
| **output dataset(s)** | **name :** evaluations.csv |
| **tool** | **name**: MySQL<br>**location**: /usr/share/mysql |
| **what should be done** | Extracting QualisBrasil from Legacy System, using a configuration file. Converting data to CSV. |
| **command** | real/QualisBrasilProject/bin/extractingFromLegacySystem.sh |

| | |
|---|---|
| **step (2)** | ldwStep_extracting_qualisbrasil_applying_sparqlify |
| **task classification** | [create a vocabulary that fits your data]<br>[reuse existing RDF vocabularies] |
| **input dataset(s)** | **name :** evaluations.csv |
| **output dataset(s)** | **name :** evolutions.nt |
| **tool** | **name**: Sparqlify<br>**location**: /usr/share/lib/sparqlify/ |
| **what should be done** | Using Sparqlify and a configuration file tp convert data to N-Triples. |
| **command** | real/QualisBrasilProject/bin/applyingSparqlify.sh |
| **tool configuration** | **name :** mappingQualis.sml |

| | |
|---|---|
| **step (3)** | ldwStep_storing_qualisbrasil |
| **task classification** | [ensure the high availability of the data for real-time applications by using state-of-the-art triplestores]<br>[automatize the process in a way that enables automatic storing] |
| **input dataset(s)** | **name :** QualisBrasil.nt |
| **output dataset(s)** | **name :** Qualis Brasil Graph |
| **what should be done** | In this step the file QualisBrasil.nt is uploaded to Virtuoso by executing the command based on the toolConfiguration file. |
| **command** | real/commons/savingIntoVirtuoso.sh real/Graphs/QualisBrasil/QualisBrasil.nt http://lod.unicentro.br/QualisBrasil/ |

| | |
|---|---|
| **step (4)** | ldwStep_finalMessage_qualisbrasil |
| **what should be done** | it shows a message informing the workflow is finished. |
| **command** | echo "Finito Sparqlify..." |

```

After applying the qualisConversion LDWProject, we got 698668 entities of Qualis
    journal evaluations, according the sparql query and results showing below.
\begin{lstlisting}
#Number of entities per year in the Graph
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT ?yearEvaluation count(?journal) as ?NUM_EVALUATIONS_PER_YEAR WHERE {
  ?evaluation rdf:type rdf:Class .
  ?evaluation qualis:hasJournal ?journal .
  ?evaluation qualis:hasYear ?yearEvaluation .
  ?yearEvaluation rdf:type rdf:Class .
  FILTER (str(?yearEvaluation) != "http://lod.unicentro.br/QualisBrasil/YearEvaluation"
      )
} GROUP BY ?yearEvaluation ORDER BY ?yearEvaluation
```

| yearEvaluation | NUM_EVALUATIONS_PER_YEAR |
|---|---|
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2005 | 35020 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2006 | 35020 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2007 | 35020 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2008 | 54233 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2009 | 54233 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2010 | 54233 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2011 | 107429 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2012 | 107429 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2013 | 107429 |
| http://lod.unicentro.br/QualisBrasil/YearEvaluation_2014 | 108622 |

### 5.3.2 qualisInterlinking LDWProject

| **1) PROJECT IDENTIFICATION** | |
|---|---|
| **1.1) Name** | Interlinking DBPedia to Qualis Project |
| **1.2) Creator** | Sandro Rautenberg |
| **1.3) Homepage** | http://lodunicentro.aksw.org/ |
| **1.5) Purpose(s)** | Interlinking Qualis Brasil to DBPedia. |
| **1.6) Description** | qualisInterlinking LDWProject converts the Qualis data from a 4 to a 5 Stars Open Data. |

| **2) PLANNING WORKFLOW** | |
|---|---|
| **Name** | **Planning Workflow of interlinking QualisBrasil with LIMES** |
| **What should be done** | This LDWorkflow comprises three LDWSteps: 1) uses LIMES to interlink Qualis with DBpedia data; 2) reuses the store LDWStep from qualisConversion LDWProject; and 3) provides user with feedback to the LDWorkflowExecution |
| **Precondition** | qualisConversion LDWProject must be executed. |
| **Postcondition** | Qualis data is interlinked and uploaded in a triple store. |
| **Step(s)** | |

| | |
|---|---|
| **step (1)** | ldwStep_interlinking_qualisbrasil_applying_LIMES |
| **task classification** | [define related open datasets to be interlinked]<br>[create consumption patterns which specify the data selection and the method of interlinking]<br>[automatize the interlinking process in a way that enables automatic updates] |
| **input dataset(s)** | **name :** Qualis Brasil Graph<br>**name :** DBpedia |
| **output dataset(s)** | **name :** QualisBrasil.nt |
| **what should be done** | applying LIMES and a tool configuration. |
| **command** | real/QualisBrasil_to_DBpedia_InterlinkingProject/bin/interlinkingQualisToDBPedia.sh |
| **tool configuration** | **name :** linkingQualisToDBpedia.xml |

| | |
|---|---|
| **step (2)** | ldwStep_storing_qualisbrasil |
| **task classification** | [ensure the high availability of the data for real-time applications by using state-of-the-art triplestores]<br>[automatize the process in a way that enables automatic storing] |
| **input dataset(s)** | **name :** QualisBrasil.nt |
| **output dataset(s)** | **name :** Qualis Brasil Graph |
| **what should be done** | In this step the file QualisBrasil.nt is uploaded to Virtuoso by executing the command based on the toolConfiguration file. |
| **command** | real/commons/savingIntoVirtuoso.sh real/Graphs/QualisBrasil/QualisBrasil.nt http://lod.unicentro.br/QualisBrasil/ |

| | |
|---|---|
| **step (3)** | ldwStep_finalMessage_interlinking_qualisbrasil |
| **what should be done** | only shows a message |
| **command** | echo "Finito LIMES..." |

After applying the qualisInterlinked LDWProject, we got 4036 entities of Qualis journals interlinked to the Linked Data cloud.

```
#Number of entities interlinked to DBpedia in 01-04-2015.
PREFIX qualis:   <http://lod.unicentro.br/QualisBrasil/>
PREFIX dc:   <http://purl.org/dc/elements/1.1/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT count(?journal) AS ?NUM_RESOURCES_INTERLINKED_TO_DBPEDIA WHERE {
  ?dbpedia owl:sameAs  ?journal .
  {
    SELECT DISTINCT ?journal WHERE {
      ?s qualis:hasJournal ?journal .
    }
  }
}
```

| NUM_RESOURCES_INTERLINKED_TO_DBPEDIA |
|---|
| 4036 |

# 6 Other results

Here we present the another results by developing LDWPO. These are tools developed to support the execution of real-world use cases of applying the ontology. Also, a real-world RDF dataset is available as Linked Open Data representing Qualis data, a scientometric dataset used by the Brazilian Scientific Community (see the VoID description of this dataset).

## 6.1 reportgen Tool

We developed a report tool for generating an LDWProject document. This tool is named reportgen Tool and is available at `https://github.com/AKSW/ldwpo/tree/master/tools`).

To use reportgen go to tool/reportgen/ folder and run “$ java -jar reportgen.jar".

The command will output a help message.

## 6.2 ldStackWorkflow Tool

We developed a workflow tool for executing an LDWProject in an automatic way. This tool is named ldStackWorkflow Tool and is available at `https://github.com/AKSW/ldwpo/tree/master/tools`).

To use ldStackWorkflow go to tool/ldStackWorkflow/ folder and run “$ java -jar ldStack-Workflow.jar".

The command will output a help message.

## 6.3 Qualis Brasil Linked Open Data dataset

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dctypes: <http://purl.org/dc/dcmitype/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix freq: <http://purl.org/cld/freq/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix sio: <http://semanticscience.org/resource/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix uniprot: <http://purl.uniprot.org/uniprot/> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://lod.unicentro.br/QualisBrasil/>
  dcterms:title "Qualis Brasil" ;
  dcterms:description "Qualis Brasil encompasses indirect scores for research papers in
       journals, according to 48 knowledge fields, such as computer science, chemistry
      , medicine, among others. It is used to rank post-graduate programs, research
      proposals, or individual research scholarships. A typical entry of Qualis
      consists of ISSN, journal name, knowledge field, and qualified journal score.
      Moreover, the journal can be related to one or more scores, based on the
      relevance of the journal to the knowledge field." ;
  dcterms:publisher  <Sandro_Rautenberg>;
  dcterms:issued "2015-04-01T00:00:00.000+01:00"^^xsd:dateTime ;
  dcterms:license <http://creativecommons.org/publicdomain/zero/1.0/> ;
  dcterms:subject <http://dbpedia.org/page/Scientometrics> ;
  dcterms:subject <http://dbpedia.org/page/Bibliometrics>;
  dcterms:source <http://qualis.capes.gov.br/webqualis/principal.seam> ;
  dcterms:accrualPeriodicity freq:annual ;

  void:sparqlEndpoint <http://lodkem.led.ufsc.br:8890/sparql> ;
  void:exampleResource <http://lod.unicentro.br/QualisBrasil/Journal_0004-5411
      _KnowledgeField_2_YearEvaluation_2014_Qualis_A1> ;
  void:exampleResource <http://lod.unicentro.br/QualisBrasil/Journal_0004-5411> ;
  void:exampleResource <http://lod.unicentro.br/QualisBrasil/KnowledgeField_2> ;
  void:exampleResource <http://lod.unicentro.br/QualisBrasil/Qualis_A1> ;
  void:exampleResource <http://lod.unicentro.br/QualisBrasil/YearEvaluation_2014> ;
  void:triples 3590448 ;
  void:entities 698669 ;
  void:properties 9 ;
a void:Dataset ;.

<Sandro_Rautenberg>
  rdfs:label "Sandro Rautenberg" ;
  foaf:mbox <mailto:srautenberg@unicentro.br> ;
  foaf:homepage <http://buscatextual.cnpq.br/buscatextual/visualizacv.do?metodo=
      apresentar&id=K4700255Z6>;  a foaf:Person; .
```

# 7 License for using LDWPO

This ontology will be published under the license “Attribution-ShareAlike CC BY-SA".

## Acknowledgment

This work was supported by the Brazilian Federal Agency for the Support and Evaluation of Graduate Education (CAPES/Brazil), under the program Sciences without Borders (Process number - 18228/12-7).

## References

[1] Sören Auer, Lorenz Bühmann, Christian Dirschl, Orri Erling, Michael Hausenblas, Robert Isele, Jens Lehmann, Michael Martin, Pablo N. Mendes, Bert van Nuffelen, Claus Stadler, Sebastian Tramp, and Hugh Williams. Managing the life-cycle of linked data with the LOD2 stack. In *Proceedings of International Semantic Web Conference (ISWC 2012)*, 2012.

[2] Sören Auer. Introduction to lod2. In Sören Auer, Volha Bryl, and Sebastian Tramp, editors, *Linked Open Data – Creating Knowledge Out of Interlinked Data*. Springer-Verlag, 2014.

[3] Asunción Gomez-Perez, Mariano Fernandez-Lopez, and Oscar Corcho, editors. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web (2nd edition)*. Springer-Verlag, Heidelberg, 2007.

[4] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report SMI-2001-0880, Stanford University School of Medicine, 2001. This guide describes a common methodology for ontology-development based on declerative frame-based systems. Upshot: there is no single correct ontology for any domain.

[5] J.F. Sowa and John A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992.

[6] York Sure and Rudi Studer. On-To-Knowledge methodology. In John Davies, Dieter Fensel, and Frank van Harmelen, editors, *On-To-Knowledge: Semantic Web enabled Knowledge Management*, chapter 3, pages 33–46. J. Wiley and Sons, 2002.