Linked Data Workflow Project Ontology – The Technical Report –

Sandro Rautenberg[1], Ivan Ermilov[2], and Edgard Marx[2]

[1] DECOMP/UNICENTRO, Computer Science Department, Midwestern State University, Paraná, Brazil, srautenberg@unicentro.br
[2] AKSW, Institute of Computer Science, University of Leipzig, Leipzig, Germany
{iermilov, marx}@informatik.uni-leipzig.de

Introduction

This report describes the development of Linked Data Workflow Project Ontology, the LDWPO.

The applied process is the result of the combination of some methodological artifacts and best practices from On-to-Knowledge [5], METHONTOLOGY [2], and Ontology Development 101 Guide [3].

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.

As result, we define an ontology development process based on four main activities to perform: ontology specification; knowledge acquisition; ontology implementation; and ontology verification. The (see Figure 1) represents this process, relating all inner tasks for each activity.

In the next sections, we discuss briefly each of these activities, presenting the main artifacts used to develop our ontology approach.

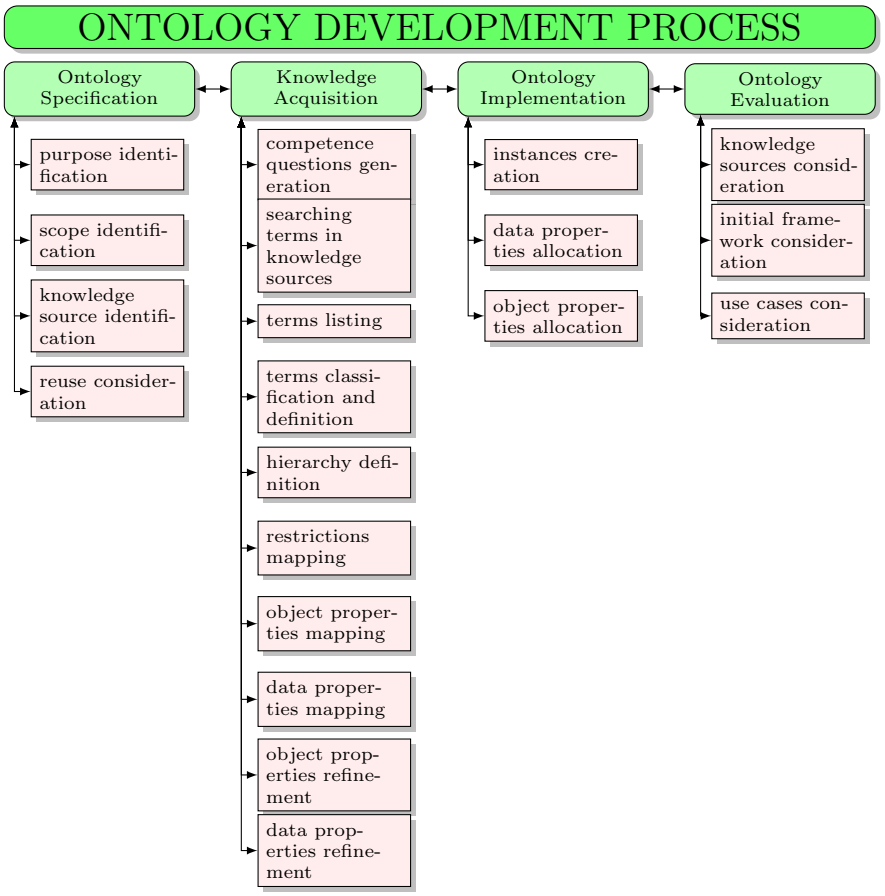

Fig. 1. Ontology development process applied to develop LDWPO.

1 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.

1.1 LDWPO scope

The scope of this ontology approach addresses some provenance and reproducibility issues in Linked Data Workflow Projects. It models simple piece of knowledge about projects, process, people, stages, steps, resources, and workflow for documenting the producing of linked datasets in a systematic way. In an engineering point of view, this supports the activities of maintainability of RDF dataset over the time, considering the Linked Data Lifecycle [1].

1.2 LDWPO purpose

Managing the lifecycle of RDF dataset maintaining. Generating useful documents for reproducing the RDF dataset maintaining workflows. Mediating the use of Linked Data Stack technologies for (semi)automatized execution of workflows.

1.3 Knowledge source identification

1. ***Dublin Core* (DC)[3] -** is a simple and powerful vocabulary used to describe web resources, promoting interoperability in Linked Data environments.
2. ***Description of a Project* (DOAP)[4] -** 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] `http://purl.org/dc/terms/`
[4] `https://github.com/edumbill/doap`

3. ***Friend of a Friend*** **(FOAF)**[5] **-** 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)**[6] **-** 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)**[7] **-** 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)**[8] **-** 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)**[9] **-** 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***[10] **-** 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.

[5] `http://xmlns.com/foaf/0.1/`
[6] `http://purl.org/net/opmv/ns`
[7] `http://www.w3.org/ns/prov#`
[8] `http://purl.org/spar/pwo`
[9] `http://aksw.org/About.html`
[10] `http://dbpedia.org/About`

2 Knowledge Acquisition activity

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.
- **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.

The main artifact of this activity presented below.

2.1 List of competence questions to be used to verify the ontology

Below, we list the competence questions organized according the Zachmann Framework [4].

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 are the tools employed in this stage?
5. What are the algorithms implemented in this tool?
6. What are the parameters we need set up in this algorithm?
7. What are the parameter's values for this tool in this stage?
8. What are the input datasets of this stage?
9. What are the datasets of this project?
10. What are the stages of this project?
11. What is the done transformation over the dataset in this stage?
12. What is the output dataset of this stage?
13. What are the done steps in this stage?
14. What is the previous step of this step?
15. What is the next step of this step?
16. What is the format of the dataset?
17. What are the project's goals?
18. What could be the news tools used in this stage?
19. What is the license of the dataset?

Dimension Where

20. Where is the input dataset stored?
21. Where is the output dataset stored?

Dimension When

22. When will be the dataset updated?
23. When was the dataset updated last time?
24. When was the step performed?
25. When did the workflow start?
26. When did the workflow finish?

Dimension Why

– *No questions were made.*

Dimension Who

27. Who are the people that contribute in this project?
28. Who is the responsible for this stage?

Dimension How

29. How is the input dataset stored?
30. How is the output dataset stored?

2.2 List of useful resources from knowledge sources

Considering the knowledge sources, the Table 1 lists potential resources for reusing.

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.

2.3 Definitions of classes, data properties, object properties of LDWPO

List of classes

1. **Artifact.** Class that represents the resources (*Datasets* and *Tools*) used when a *Step* is performed. *[owl:equivalentClass - opmv:Artifact]*
2. **Dataset.** Subclass of *Artifact* that represents a collection of data. Related to a *Step*, it could be an input (before execution) or an output (after execution) of *Step* execution. *[owl:equivalentClass - dc:Dataset, opmv:Artifact, and prov:Collection]*
3. **RDFDataset.**

4. **Homepage.**
5. **Report.**
6. **Tool.** subclass of *Artifact* that represents a computational implementation of one or more *Algorithm(s). [owl:equivalentClass - dc:Software, opmv:Artifact, and opmv:SoftwareAgent]*
7. **ToolConfiguration.**
8. **Condition.**
9. **FileFormat.** Subclass of *Parameter* that represents the encode format of a *Dataset*, such as SQL, CSV, or RDF. *[owl:equivalentClass - dc:FileFormat]*
10. **KnowledgeBody.**
11. **Project.** Class that represents an endeavour of creating or maintaining a linked dataset. It involves a methodological *Process* and an executable *Workflow* to perform it. *[owl:equivalentClass - prov:Plan, doap:Project, and foaf:Project]*
12. **Execution.**
13. **LDWorkflowExecution.**
14. **LDWStepExecution.**
15. **Method.**
16. **Process.** Class that represents some concise set of *Stages* performed to create or maintain a linked dataset.
17. **Stage.** Class that represents a concise set of *Steps* to be performed on *Dataset(s)* and resulting an intermediary or final result. *[owl:equivalentClass - opmv:Process, and prov:Plan]*
18. **Task.**
19. **Plan.**
20. **LDWStep.**
21. **LDWorkflow.** class that represents a sequence of connected *Step(s)* to produce or maintain a linked dataset. *[owl:equivalentClass - prov:Plan, opmv:Process, and pwo:Workflow]*
22. **License.** Subclass of *Parameter* that represents the type of access permission to a *Dataset*. *[owl:equivalentClass - dc:LicenseDocument]*
23. **Location.** Subclass of *Parameter* that represents the path to access a *Dataset*. *[owl:equivalentClass - dc:Location]*
24. **Message.**
25. **Person.** Class that represents people involved in a part of or whole *Project*, performing the roles of contributor or creator. *[owl:equivalentClass - dc:Agent, prov:Person, opmv:Agent, and foaf:Person]*
26. **Status.**
27. **Step.** Class that represents an atomic executable unit in a *Workflow* and a *Stage*. It involves some *Dataset(s)*, employment of one *Tool*, *Parameter(s)*, and *ParametersValue(s)*. It produces *Dataset(s)* as output. It is also associated to a *Person* (object property *contributor*) and when existing next and/or previous *Step(s). [owl:equivalentClass - prov:Activity, opmv:Process, pwo:Step, dc:Event]*

List of object properties

1. **contributor.** object property that points to a *Person* who performed the Step. Combined with *workflow* property, it can point out the list of *Contributors* of a *Project*. As restrictions, one *Step* can be performed by one *Contributor(s)*. And one *Contributor* can be related to one or more *Step(s)*. *[owl:equivalentProperty - opmv:wasPerformedBy, doap:developer, opmv:wasControlledBy, foaf:maker, dc:contributor, and prov:wasAssociatedWith]*
2. **creator.** object property that points to *Person* who created it. As restrictions, one *Project* is created by one *Creator*. And one *Creator* can be related to one or more *Project(s)*. *[owl:equivalentProperty - doap:maintainer, dc:creator, foaf:maker]*
3. **fileFormat.** object property that points to the *FileFormat* of a *Dataset*. As restrictions, one *FileFormat* is related to one or more *Dataset(s)*. And a *Dataset* is related to a single *FileFormat*. *[owl:equivalentProperty - dc:format]*
4. **firstLdwStep.** object property that points to *Step* that starts the execution of the whole *Workflow*. As restrictions, one *Workflow* has one initial *Step*. And one *Step* can be related to one *Workflow*. *[owl:equivalentProperty - pwo:hasFirstStep]*
5. **firstLdwStepExecution.**
6. **homepage.**
7. **inputDataset.** object property that points to the *InputDataset(s)* used by a *Step*. Combining with *step*, it can point out the list of *InputDatasets* used in a *Stage*. As restrictions, one *Step* has one or more *InputDataset(s)*. And one *InputDataset* is related to one or more *Step(s)*. *[owl:equivalentProperty - pwo:needs, opmv:used, and prov:used]*
8. **ldWorkflow.**
9. **ldWorkflowExecution.**
10. **ldwStep.**
11. **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 *Dataset(s)*. *[owl:equivalentProperty - dc:rights]*
13. **location.** object property that points to the physical location of the *Dataset*, that is URL or local filesystem path. *[owl:equivalentProperty - prov:atLocation]*
14. **message.**
15. **nextStep.** an irreflexive and asymmetric object property that points to the next *Step* to perform of the current *Step*. It has *previousStep* as an inverse property. As restriction, one *Step* has only one next *Step*. *[owl:equivalentProperty - pwo:hasNextStep]*
16. **outputDataset.** object property that points to the *OutputDataset(s)* transformed by a *Step*. Combining with *step*, it can point out the list of *OutputDatasets* processed in a *Stage*. As restrictions, one *Step* has only one *OutputDataset* And one *OutputDataset* is related to one *Step*. *[owl:equivalentProperty - pwo:produces]*
17. **postcondiction.**

18. **preCondiction.**
19. **previousStep.**
20. **previousStep.** an irreflexive and asymmetric object property that points to the previous *Step* of the current *Step*. It has *nextStep* as an inverse property. As restriction, one *Step* has only one previous *Step*. *[owl:equivalentProperty - pwo:hasPreviousStep]*
21. **report.**
22. **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)*.
23. **status.**
24. **step.** object property that points to the *Step(s)* of a *Stage*. As restrictions, one *Stage* is related to one or more *Step(s)*. And one *Step* is related to a single *Stage*. *[owl:equivalentProperty - pwo:hasStep]*
25. **task.**
26. **tool.** object property that points to the *Tool(s)* employed in a *Step*. As restrictions, one *Step* can employ one *Tool*. And one *Tool* is related to one or more *Step(s)*. *[owl:equivalentProperty - pwo:needs, and opmv:used]*
27. **toolConfiguration.**

List of data properties

1. **command.** a functional data property that contains the console command to run during a *Step*. As restriction, *description* accepts only xsd:String values.
2. **description.** a functional data property used to describe a *Project*. As restriction, *description* accepts only xsd:String values. *[owl:equivalentProperty - dc:description, and doap:description]*
3. **endedDate.** a functional data property that expresses when a *Workflow* was finished. As restriction, *finalDate* accepts only xsd:DateTime values. *[owl:equivalentProperty - opmv:wasEndedAt, and prov:endedAtTime]*
4. **goal.** a non-functional data property that expresses the set of goals of a *Project*. As restriction, *goal* accepts only xsd:String values. *[owl:equivalentProperty - doap:implements]*
5. **name.** a functional data property used by *Algorithm*, *Dataset*, *License*, *Parameter*, *Person*, *Project*, *Stage*, *StoredFormat*, and *Tool* to name a instance. As restriction, *name* accepts only xsd:String values. *[owl:equivalentProperty - foaf:name]*
6. **startedDate.** a functional data property that expresses when a *Workflow* was started. As restriction, *initialDate* accepts only xsd:DateTime values. *[owl:equivalentProperty - dc:date, dc:created, doap:created, prov:startedAtTime, and opmv:wasStartedAt]*
7. **value.** a functional data property that encodes the value of a ParameterValue in xsd:String format.

2.4 Class hierarchy

Figure 2.

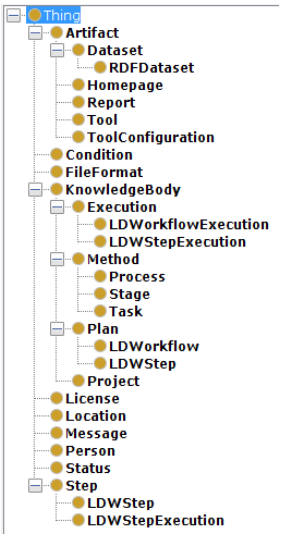

Fig. 2. LDWPO class hierarchy.

2.5 Object properties refinement

The Table 2 lists all objects properties relating them to the domain and ranges.

2.6 Data properties refinement

The Table 3 lists all data properties relating them to the domain and ranges.

2.7 Class diagram

Figure 3.

Object Property	Domain	Range
hasAlgorithm	Tool	Algorithm
hasContributor	Step	Person
hasCreator	Project	Person
hasDefaultValue	Parameter	DefaultValue
hasFileFormat	Dataset	FileFormat
hasFirstStep	Workflow	Step
hasInputDataset	Step	Dataset
hasLicense	Dataset	License
hasLocation	Dataset	Location
hasNextStep	Step	Step
hasOutputDataset	Step	Dataset
hasParameter	Algorithm, Artifact, Step	Parameter
hasParameterValue	Artifact, Parameter, Step	ParameterValue
hasPreviousStep	Step	Step
hasStage	Process	Stage
hasStep	Stage, Workflow	Step
hasTool	Step	Tool
hasWorkflow	Project	Workflow

Table 2. Object properties - domains and ranges.

Data Property	Domain	Range
hasCommand	Step	String
hasDescription	Project	String
hasFinalDate	Workflow	Datetime
hasGoal	Project	String
hasHomepage	Project	String
hasInitialDate	Workflow	Datetime
hasLastUpdatedDate	Dataset	Datetime
hasName	Algorithm, Dataset, License, Parameter, Person, Project, Stage, StoredFormat, Tool	String
hasNextUpdateDate	Dataset	Datetime
hasPerformedDate	Step	Datetime
hasScript	Step	String
hasShortDescription	Stage, Step, Workflow	String
hasValue	ParameterValue	String

Table 3. Data properties - domains and ranges.

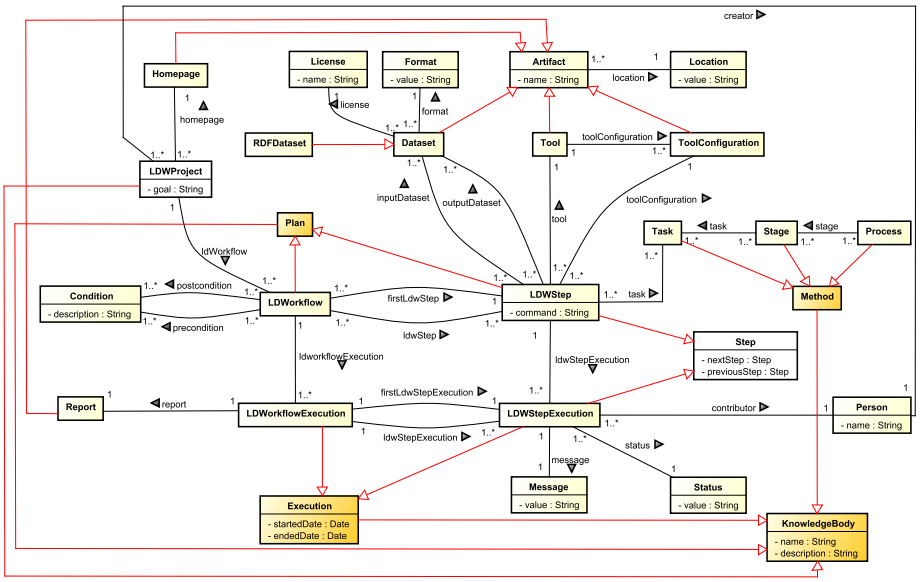

Fig. 3. Representing LDWPO, highlighting the abstract concepts Knowledge Body, Method, Plan, and Execution.

3 Ontology Implementation activity

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 resulting artifact, we point to the ontologi (available at `http://domain.com`)

4 Ontology Verification activity

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.
- **The initial framework consideration** confirms the accuracy and completeness of the ontology, taking into account the purpose, scope, and mainly, competence questions.
- **Use cases consideration** inspects the usefulness of the ontology by simulating some scenarios.

4.1 Knowledge source consideration

4.2 Initial framework consideration

To confirm the accuracy and completeness of LDWPO, following we present the competence questions coded as SPARQL queries.

```
01. 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
```

4.3 Use cases consideration

References

1. 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.
2. 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.

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

textbfResource	**equivalence**	**Knowledge resource**
dc:Agent	owl:equivalentClass	Person
dc:Dataset	owl:equivalentClass	Dataset
dc:Event	owl:equivalentClass	Step
dc:FileFormat	owl:equivalentClass	FileFormat
dc:LicenseDocument	owl:equivalentClass	License
dc:Software	owl:equivalentClass	Tool
doap:Project	owl:equivalentClass	Project
foaf:Person	owl:equivalentClass	Person
foaf:Project	owl:equivalentClass	Project
opmv:Agent	owl:equivalentClass	Person
opmv:Artifact	owl:equivalentClass	Artifact
opmv:Process	owl:equivalentClass	Stage
opmv:Process	owl:equivalentClass	Step
opmv:Process	owl:equivalentClass	Workflow.
opmv:SoftwareAgent	owl:equivalentClass	Tool
prov:Activity	owl:equivalentClass	Step
prov:Collection	owl:equivalentClass	Dataset
prov:Location	owl:equivalentClass	Location
prov:Person	owl:equivalentClass	Person
prov:Plan	owl:equivalentClass	Project
prov:Plan	owl:equivalentClass	Stage
prov:Plan	owl:equivalentClass	Workflow.
pwo:Step	owl:equivalentClass	Step
pwo:Workflow	owl:equivalentClass	Workflow.
dc:contributor	owl:equivalentProperty	hasContributor
dc:creator	owl:equivalentProperty	hasCreator
dc:format	owl:equivalentProperty	hasFileFormat
dc:rights	owl:equivalentProperty	hasLicense
doap:developer	owl:equivalentProperty	hasContributor
doap:maintainer	owl:equivalentProperty	hasCreator
foaf:maker	owl:equivalentProperty	hasContributor
foaf:maker	owl:equivalentProperty	hasCreator
opmv:used	owl:equivalentProperty	hasInputDataset
opmv:used	owl:equivalentProperty	hasTool
opmv:wasControlledBy	owl:equivalentProperty	hasContributor
opmv:wasPerformedBy	owl:equivalentProperty	hasContributor
prov:atLocation	owl:equivalentProperty	hasLocation
prov:used	owl:equivalentProperty	hasInputDataset
prov:wasAssociatedWith	owl:equivalentProperty	hasContributor
pwo:hasFirstStep	owl:equivalentProperty	hasFirstStep
pwo:hasNextStep	owl:equivalentProperty	hasNextStep
pwo:hasPreviousStep	owl:equivalentProperty	hasPreviousStep
pwo:hasStep	owl:equivalentProperty	hasStep
pwo:needs	owl:equivalentProperty	hasInputDataset
pwo:needs	owl:equivalentProperty	hasTool
pwo:produces	owl:equivalentProperty	hasOutputDataset
dc:created	owl:equivalentProperty	hasInitialDate
dc:date	owl:equivalentProperty	hasInitialDate
dc:description	owl:equivalentProperty	hasDescription
dc:description	owl:equivalentProperty	hasShortDescription
doap:created	owl:equivalentProperty	hasInitialDate
doap:description	owl:equivalentProperty	hasDescription
doap:homepage	owl:equivalentProperty	hasHomepage
foap:homepage	owl:equivalentProperty	hasHomepage
doap:implements	owl:equivalentProperty	hasGoal
foaf:name	owl:equivalentProperty	hasName
opmv:wasEndedAt	owl:equivalentProperty	hasFinalDate
opmv:wasGeneratedAt	owl:equivalentProperty	hasLastUpdatedDate
opmv:wasPerformedAt	owl:equivalentProperty	hasPerformedDate
opmv:wasStartedAt	owl:equivalentProperty	hasInitialDate
prov:endedAtTime	owl:equivalentProperty	hasFinalDate
prov:startedAtTime	owl:equivalentProperty	hasInitialDate

Table 4. List of resources and its equivalence to another knowledge sources.