An Algorithm, Implementation and Execution Ontology Design Pattern

Agnieszka Ławrynowicz[1], Diego Esteves[2], Panče Panov[3], Tommaso Soru[2], Sašo Džeroski[3] and Joaquin Vanschoren[4]

[1] Faculty of Computing, Poznan University of Technology, Poznan, Poland
[2] AKSW, University of Leipzig, Germany
[3] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
[4] Eindhoven University of Technology, The Netherlands

Abstract. This paper describes an ontology design pattern for modeling algorithms, their implementations and executions. This pattern is derived from the research results on data mining/machine learning ontologies, but is more generic. We argue that the proposed pattern will foster the development of standards in order to achieve a high level of interoperability among in silico scientific experiments. We describe the intent of the pattern, provide competency questions and the pattern formalization. We also present an example instantiation of the pattern in the machine learning domain.

Keywords: ODP, machine learning, data mining, interoperability

1 Introduction

Artificial Intelligence has become an important area for data scientists in research and business contexts. A wide range of algorithms, implementations and tools have emerged to deal with particular problems, contributing greatly to the progress and development of its sub-areas, such as Data Mining (DM) and Machine Learning (ML). These breakthroughs, however, have exposed a new gap: *how to properly represent and interchange this information among systems and researchers?*, for instance, how to differentiate *Support Vector Machines (SVM)* implementations among existing tools. To address this problem (as well as other interoperability issues), state of the art data mining and machine learning ontologies [7, 10, 14] and vocabularies [5] have been proposed.

In this paper, we describe an ontology design pattern (ODP) for modeling algorithms, their implementations and executions, as well as produced outputs. We aim at defining a logical interface to guide the design process of representing the generic triple defined by *input, algorithm and output.* One advantage of this conceptual alignment is that it allows us to formalize conceptual definitions, facilitating the metadata interchange process and fostering reproducible research. Furthermore, we directly collaborate with a W3C Community Group[5]

[5] `https://www.w3.org/community/ml-schema/`

which aims to define an upper level ontology which maps metadata from various existing ML/DM schemata. In particular, all of the co-chairs of the group are co-authors of this pattern paper. Finally, we argue that the proposed pattern will stimulate the development of standards in order to achieve high level of interoperability among scientific experiments. Thus, the proposed template provides guidance for supporting further developments of schemata that represent this flow, ensuring their logical alignment.

This paper is structured as follows: we propose the design pattern in Section 2, and illustrate its use in Section 3. Next, we describe how it aligns with PROV-based vocabularies in Section 4. Section 5 discusses the need and usefulness of the proposed pattern, and Section 6 concludes.

2 The Algorithm-Implementation-Execution Ontology Design Pattern

The Algorithm-Implementation-Execution Design Pattern[6] is depicted in Fig 1.

The Algorithm class represents any algorithm regardless its software implementation. Implementation is an executable implementation of an algorithm, a script, or a workflow (composite algorithm). The Execution class is an execution of an implementation on a machine (computer). It is limited in time (has a start and end point), and can be successful or failed.

Task is a formal description of a process that needs to be completed (e.g. based on inputs and outputs), and any piece of work that needs to be addressed.

Input and Output are information entities, where the former is an input to an Execution (e.g., some data), and the latter is an output of the Execution (e.g., some transformed data, the result of some computations etc.).

Parameter is a parameter of an implementation which is set before its execution. It is differentiated from Input in that it is a variable. ParameterSetting is an entity which connects a parameter and its value that is being set before an implementation execution.

2.1 Intent

The intent of the design pattern is to model algorithm specifications, their implementations and executions. This includes also the parameters of implementations, settings of these parameters for a specific execution, as well as the inputs that the execution consumes (e.g., data) and the outputs it produces (e.g., models, reports).

2.2 Competency Questions

The competency questions for the pattern were selected and refined based on the previous research on ML/DM ontologies as well as discussed during the OpenML

[6] `http://ontologydesignpatterns.org/wiki/Submissions:AlgorithmImplementationExecution`

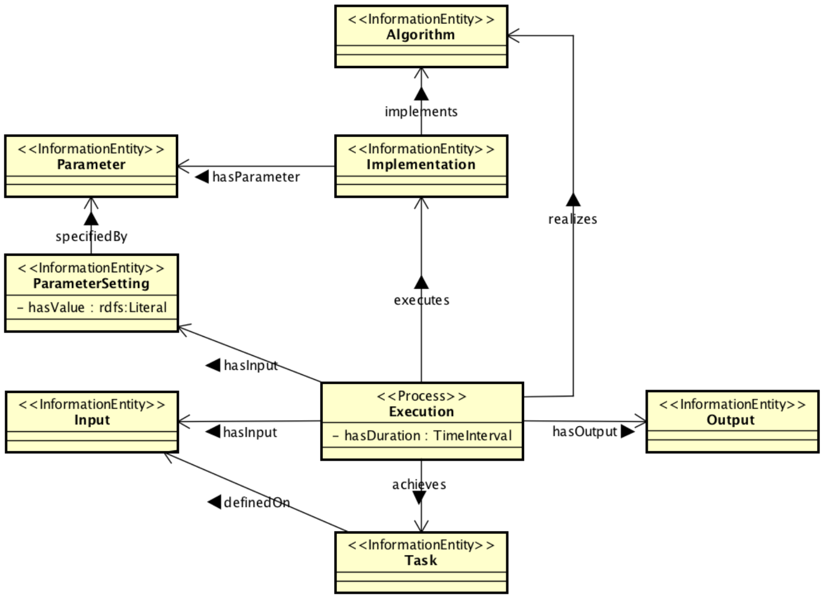

Fig. 1. The Algorithm-Implementation-Execution Ontology Design Pattern
.

Hackathon in 2016 in Lorentz Center, The Netherlands[7] in which several of the co-authors of the pattern participated. They are as follows:

- Which algorithm is implemented by this implementation?
- What are the implementations of this algorithm?
- Which implementation is executed?
- What are the parameters of this implementation?
- What are the parameter settings of particular parameters in this execution?
- What is the input to this implementation execution?
- What is the output produced by this implementation execution?
- What algorithm does this execution realize?
- What task does this execution achieves?
- What is the duration of this execution?
- What are the inputs this task is defined on?

[7] `http://openml2016.openml.org`

2.3 Pattern Formalization

Below, we show the formalization of the pattern in description logic (DL) [2], the logic underpinning the Web Ontology Language (OWL) [6]:

$$
\begin{aligned}
\mathsf{Algorithm} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{Implementation} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{Implementation} &\sqsubseteq \exists \mathsf{implements}.\mathsf{Algorithm}\\
\mathsf{Implementation} &\sqsubseteq \exists \mathsf{hasParameter}.\mathsf{Parameter}\\
\mathsf{Execution} &\sqsubseteq \mathsf{Process}\\
\mathsf{Execution} &\sqsubseteq \exists \mathsf{hasInput}.\mathsf{ParameterSetting}\\
\mathsf{Execution} &\sqsubseteq \exists \mathsf{realizes}.\mathsf{Algorithm}\\
\mathsf{Execution} &\sqsubseteq \exists \mathsf{achieves}.\mathsf{Task}\\
\mathsf{Execution} &\sqsubseteq \exists \mathsf{hasDuration}.\mathsf{TimeInterval}\\
\mathsf{Parameter} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{ParameterSetting} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{ParameterSetting} &\sqsubseteq \exists \mathsf{specifiedBy}.\mathsf{Parameter}\\
\mathsf{ParameterSetting} &\sqsubseteq \exists \mathsf{hasValue}.\mathsf{rdfs:Literal}\\
\mathsf{Input} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{Output} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{Task} &\sqsubseteq \mathsf{InformationEntity}\\
\mathsf{Task} &\sqsubseteq \exists \mathsf{definedOn}.\mathsf{Input}\\
\top &\sqsubseteq \forall \mathsf{hasInput}.\mathsf{Input}\\
\top &\sqsubseteq \forall \mathsf{hasOutput}.\mathsf{Output}
\end{aligned}
$$

We deliberately keep the axiomatization minimalistic to increase interoperability between existing and future ontologies. For instance, existing, mentioned ML/DM ontologies have bigger coverage and often more dense formalization, and have been built with different purposes on mind.

3 Example Usage: Machine Learning Domain

In this section, we first illustrate our pattern with a simple scenario from the ML domain with an example derived from the OpenML portal [15], after which we discuss how the proposed pattern occurs in existing ML/DM ontologies and schemas.

3.1 Example Scenario

Consider a scenario in the ML domain (illustrated in Figure 2). The scenario deals with an ML task realization based on an example derived from the OpenML portal. The referenced individuals can easily be looked up online.[8]

[8] For instance, run 100241 can be found on http://www.openml.org/r/100241.

ML Task `:task29` is a supervised classification task defined on the dataset `:credit-a`. This task is achieved by the Execution `:run100241` which executes the Implementation `:wekaLogistic` of the Algorithm `:logisticRegression` which this execution realizes.

The Implementation `:wekaLogistic` has five hyperparameters (Parameter): `:wekaLogisticC`, `:wekaLogisticDoNotCheckCapabilities`, `:wekaLogisticM`, `:wekaLogisticOutputDebugInfo`, `:wekaLogisticR`. The values of two of these hyperparameters are set. The hyperparameter `:wekaLogisticM` has value set to -1 (expressed via the ParameterSetting `:wekaLogisticMSetting29`), and the hyperparameter `:wekaLogisticR` that has its value set to `"1.0E-8"^^xsd:float` (expressed via the ParameterSetting `:wekaLogisticRSetting29`).

The Execution `:run100241` has as Input the `:credit-a` dataset and the parameter settings and its Output is the ML model `:wekaLogisticModel100241`.

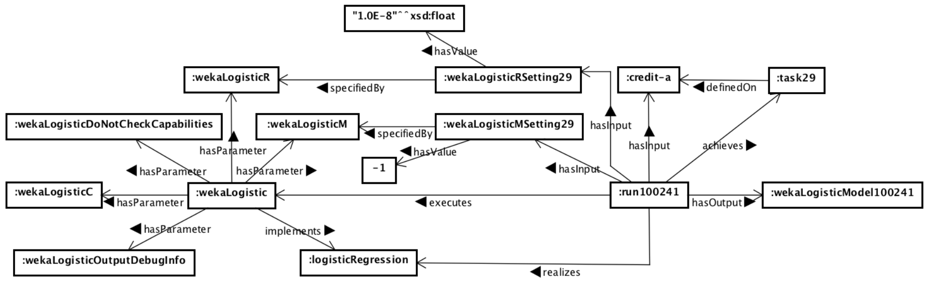

Fig. 2. Example usage of the pattern in ML domain.

3.2 Example: The occurrence of the pattern in the MEX Vocabulary

The MEX vocabulary has been designed to reuse existing ontologies (i.e., *PROV-O*[9], *Dublin Core*[10], and *DOAP*[11]) for representing basic machine learning information. The aim is not to describe a complete data-mining process, which can be modeled by more complex and semantically refined structures [4, 7, 14, 16]. Instead, MEX is designed to provide a simple and lightweight vocabulary for exchanging machine learning metadata in order to achieve a high level of interoperability as well as supporting data management for ML outcomes. Figure 4 depicts an excerpt of the interlinked classes representing the implementation of the proposed pattern.

[9] `https://www.w3.org/TR/prov-o/`
[10] `http://dublincore.org/documents/dcmi-terms/`
[11] `http://usefulinc.com/doap/`

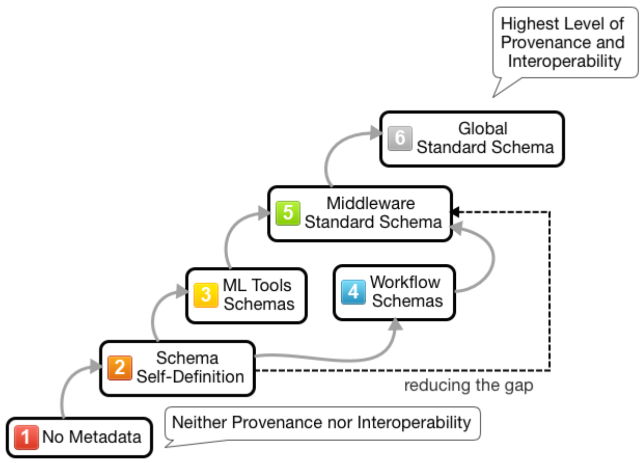
Highest Level of Provenance and Interoperability
6 Global Standard Schema
5 Middleware Standard Schema
3 ML Tools Schemas
4 Workflow Schemas
2 Schema Self-Definition
reducing the gap
1 No Metadata
Neither Provenance nor Interoperability

Fig. 3. The Evolution of the Metadata Generation Process: from no machine-readable scenarios (1) and free-style descriptions (2) until more refined structures for representing ML/DM metadata (5).

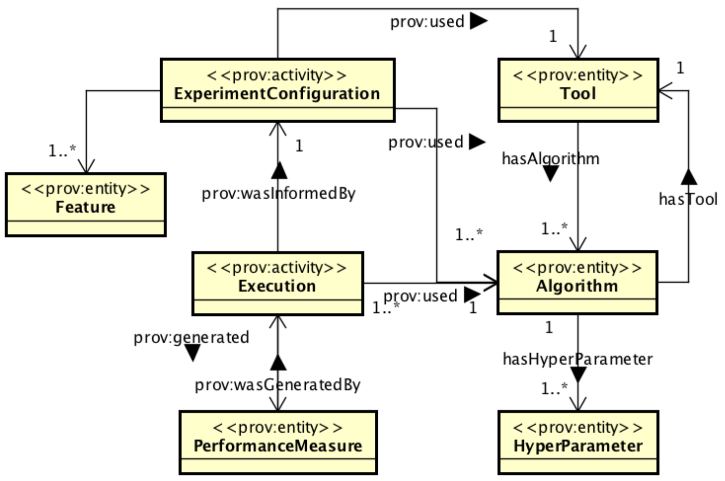

Fig. 4. An excerpt of MEX Vocabulary representing the pattern “algorithm-implementation-execution”

3.3 Example: The occurrence of the pattern in the DMOP ontology

The Data Mining OPtimization Ontology (DMOP) [7] has been developed with a primary use case in meta-mining, that is meta-learning extended to an analysis of full DM processes. At the level of both single algorithms and more complex workflows, it follows a very similar modeling pattern as our proposed ODP (see Figure 5).

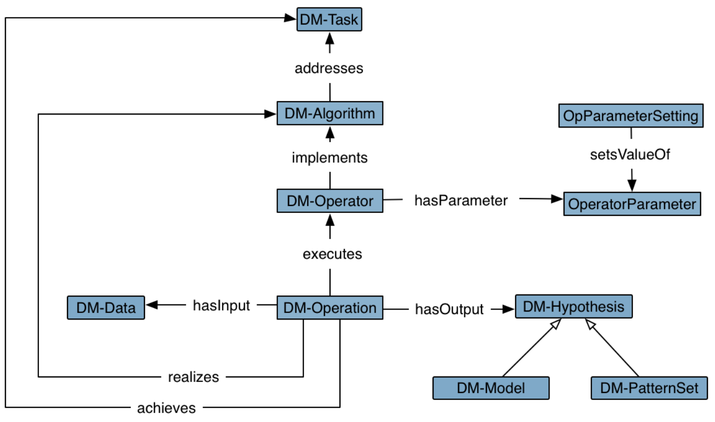

Fig. 5. The occurrence of the proposed pattern in the Data Mining OPtimization Ontology.

3.4 Example: The occurrence of the pattern in the OntoDM-core ontology

For the domain of data mining there are several developed ontologies, with the aim of providing formal descriptions of domain entities. One of the proposed ontologies is the OntoDM-core ontology [10]. In one of the preliminary versions of the ontology [9], the authors decided to align the proposed ontology with the Ontology of Biomedical Investigations (OBI) [3] and consequently with the Basic Formal Ontology (BFO) at the top level [1], in terms of top-level classes and the set of relations. That was beneficial for structuring the domain in a more elegant way and the basic differentiation of information entities, implementation entities and processual entities.

In this context, the authors proposed a horizontal description structure that includes three layers: a specification layer, an implementation layer, and an application layer. In Fig. 6, we present examples of OntoDM-core classes from all three layers. The specification layer in general contains information entities. In the domain of data mining, example classes are *data mining task* and *data mining algorithm*. The implementation layer in general contains qualities and entities that are realized in a process, such as parameters and implementations of algorithms. The application layer contains processual classes, such as the execution of the data mining algorithm. These layers were preserved in ontologies that build on OntoDM, such as Exposé [13], which underlies the OpenML platform.

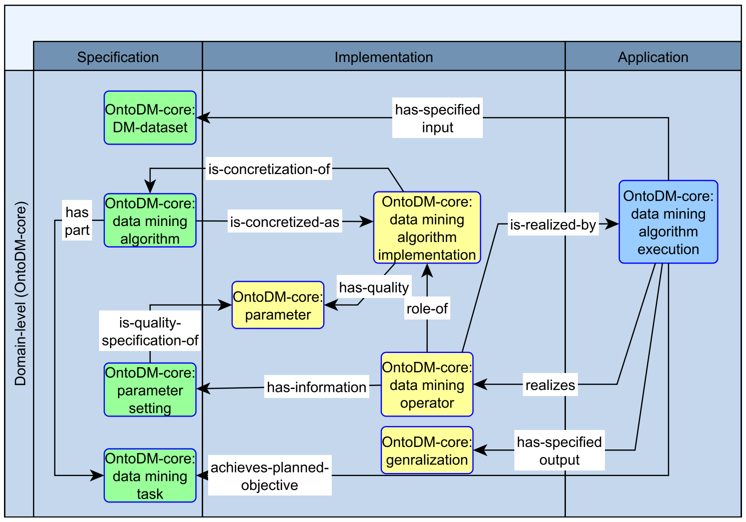
Specification
Implementation
Application
Domain-level (OntoDM-core)
OntoDM-core: DM-dataset
has-specified input
is-concretization-of
OntoDM-core: data mining algorithm
has part
is-concretized-as
OntoDM-core: data mining algorithm implementation
is-realized-by
OntoDM-core: data mining algorithm execution
has-quality
OntoDM-core: parameter
role-of
is-quality-specification-of
OntoDM-core: parameter setting
has-information
OntoDM-core: data mining operator
realizes
OntoDM-core: genralization
has-specified output
OntoDM-core: data mining task
achieves-planned-objective

Fig. 6. The horizontal three layer description structure of the OntoDM-core ontology showing the occurrence of the proposed pattern. The example classes are represented by rectangles with rounded corners. The ontological relations between classes are represented with directed labeled arrows. Unlabeled arrows represent *is-a* relations.

The idea of having all three layers represented separately in the ontology was to facilitate different uses of the ontology. For example, the specification layer can be used to reason about data mining algorithms; the implementation layer can be used for search over implementations of data mining algorithms and to compare various implementations; and the application layer can be used for searching through executions of data mining algorithms.

It is evident that there is large similarity of the design structure proposed in the OntoDM-core ontology with the proposed ontology pattern. First, there is the same differentiation between information entities, implementation entities and processual entities. Second, there is a clear mapping of the relations that are used to describe the entities. Finally, since OntoDM-core is strongly aligned with the OBI ontology, the same pattern occurs for connecting plan specifications, plans and planned processes as is illustrated in Fig. 7.

Table 1 shows mappings of the terms from the different machine learning or data mining ontologies and vocabularies to the proposed pattern.

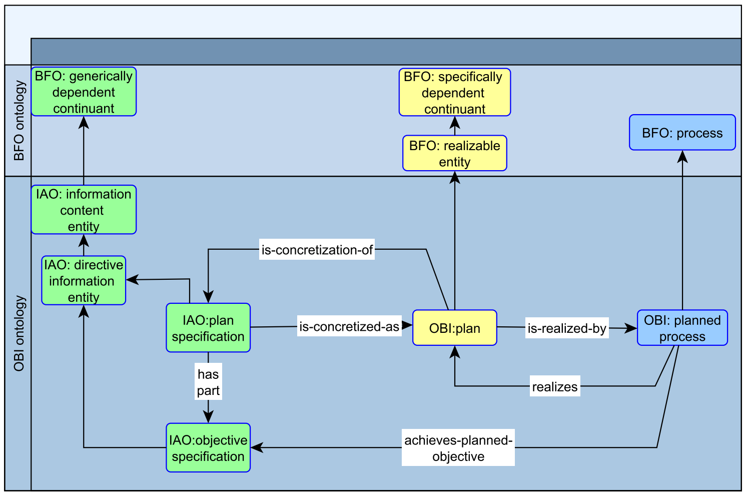
BFO ontology
OBI ontology
BFO: generically dependent continuant
BFO: specifically dependent continuant
BFO: realizable entity
BFO: process
IAO: information content entity
IAO: directive information entity
IAO:plan specification
is-concretization-of
is-concretized-as
OBI:plan
is-realized-by
OBI: planned process
realizes
has part
IAO:objective specification
achieves-planned-objective

Fig. 7. The occurrence of parts the proposed pattern in the Ontology of Biomedical Investigations for representing plan specifications, plans and planned processes.

4 Compatibility with PROV-based Vocabularies

Information about provenance has shown to play an important role in metadata descriptions. Therefore, providing compatibility with ontologies such as PROV-O has recently become strongly recommended. In this section, we show how PROV-based vocabularies can be aligned to the proposed design pattern by introducing a use case dubbed LinkLion[12], a central repository for the storage of links among resources in the Web of Data. The main goal of LinkLion is to facilitate the publication, retrieval and use of links between knowledge bases [8]. These links are collected in mappings that were generated by frameworks through link-discovery algorithms, which are usually backed by machine- and statistical-relational-learning techniques [11, 12].

How the two vocabularies align can be seen in Table 2. Since `Algorithm` and `Task` are conceptual classes, they do not map to any PROV concept. On the other hand, no information about `prov:Agent` can be modelled using our pattern.

5 Discussion

The origin of the pattern. The examples from the ML/DM domain in this paper illustrate the origin of the pattern. Due to the growth of ML/DM research, a new challenge arises pertaining to data management and knowledge sharing of scientific metadata. Even though each of the proposed approaches often provides some form of documentation, the models are often incompatible or hard to

[12] `http://www.linklion.org/`

Table 1. Mapping of terms between different ontologies/vocabularies.

Pattern	**OntoDM-core**	**DMOP** [7]	**Exposé** [14]	**KD Ontology** [16]	**KDDOnto** [4]	**MEX vocabulary** [5]
Input	DM-dataset	DataSet	dataset	dataset	dataset	Dataset
Task	data mining task	DM-Task	data mining task	data mining task	task	N/A
Output	generalization	DM-Model and DM-Pattern Set	model and pattern set	model	model	Model
Algorithm	data mining algorithm	DM-Algorithm	algorithm specification	algorithm	algorithm	Algorithm
Implementation	data mining algorithm implementation	DM-Operator	algorithm implementation	N/A	N/A	Tool
Execution	data mining algorithm execution	DM-Operation	algorithm application	algorithm execution	N/A	Execution

align. This gap affects tools and scientific software as well as vocabularies and ontologies, i.e., it has an impact horizontally and vertically. Therewith, we face an urgent need to devise patterns for representing data, especially metadata, which are considered as a great asset for gaining insight in complex world scenarios [17]. By defining design patterns and agreeing on a common schema[13] we naturally achieve a high level for experimental representation in those domains.

Whereas, in practical terms, achieving interoperability among all ML/DM tools can be considered as an utopian scenario due to business and financial issues, tools can be built based on the proposed ODP for converting metadata between any existing ML/DM schema (`eg.: mlschema.convert('mex', 'dmop', 'ttl')` or vice-versa), to help achieve this goal among schemata of ML/DM.

Relation to other content ODPs. Besides the discussed origins and the ML/DM domain where the proposed pattern is well grounded, we also note the relation of the proposed pattern to previously proposed ODPs, namely: Parameter, BasicPlan, and BasicPlanExecution.

The Parameter ODP models parameters of a concept and their values. Different than in this pattern, we introduce further reification via a class ParameterSetting since we want to talk about parameters of some implementations independently of their settings where there may be many of the latter ones (specific to particular executions of an implementation).

The BasicPlan ODP also captures Agents who participate in some actions, which is beyond our model. In our case, Implementation is most close to a plan which is then executed to achieve a particular Task (a goal). In the BasicPlan ODP those concepts are structured differently, i.e., it is the plan that defines tasks and roles of an object (e.g. of an agent). Thus, a task in the BasicPlan

[13] `https://www.w3.org/community/ml-schema/`

Table 2. Alignment between our pattern proposal, PROV-O, and the use case LinkLion. Note that every LinkLion class subsumes the corresponding PROV-O class and each PROV-O class subsumes a corresponding class from our pattern.

Pattern	**PROV-O**	**LinkLion**
Execution	prov:Activity	llont:Algorithm
Input	prov:Entity	void:Dataset
Output	prov:Entity	llont:Link, llont:Mapping
Implementation	prov:Entity	N/A
ParameterSetting	prov:Entity	N/A
hasInput	prov:used	prov:used
hasParameterSetting	prov:used	prov:used
executes	prov:used	prov:used
specifiedBy	prov:wasDerivedFrom	prov:wasDerivedFrom
hasOutput	prov:wasGeneratedBy^{-1}	prov:wasGeneratedBy^{-1}

ODP is more fine-grained and it serves to achieve a goal where the latter one is then understood more like our Task. BasicPlanExecution concerns executing a plan which involves actions and their participants.

6 Conclusions

In this paper, we have proposed the Algorithm-Implementation-Execution Ontology Design Pattern. This pattern has originated from, initially independent, research of several groups on modeling the machine learning/data mining domain. We have shown how this pattern re-appears in the ML/DM ontologies and schemas, and illustrated it with a practical examples of ML algorithm executions. However, the pattern is more generic and we have also discussed its relation to PROV and scientific interoperability in a broader scope as well as to existing ODPs.

Acknowledgements. Agnieszka Ławrynowicz acknowledges the support from the National Science Centre, Poland, within the grant number 2014/13/D/ST6/02076. Panče Panov and Sašo Džeroski acknowledge the support by the EC (FP7 framework programme) within the grant ICT-2013-612944 MAESTRA.

References

1. Arp, R., Smith, B., Spear, A.D.: Building Ontologies with Basic Formal Ontology. The MIT Press (2015)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, New York, NY, USA (2003)
3. Bandrowski, A., Brinkman, R., Brochhausen, M., Brush, M.H., Bug, B., Chibucos, M.C., Clancy, K., Courtot, M., Derom, D., Dumontier, M., Fan, L., Fostel, J.,

Fragoso, G., Gibson, F., Gonzalez-Beltran, A., Haendel, M.A., He, Y., Heiskanen, M., Hernandez-Boussard, T., Jensen, M., Lin, Y., Lister, A.L., Lord, P., Malone, J., Manduchi, E., McGee, M., Morrison, N., Overton, J.A., Parkinson, H., Peters, B., Rocca-Serra, P., Ruttenberg, A., Sansone, S.A., Scheuermann, R.H., Schober, D., Smith, B., Soldatova, L.N., Stoeckert, Jr., C.J., Taylor, C.F., Torniai, C., Turner, J.A., Vita, R., Whetzel, P.L., Zheng, J.: The ontology for biomedical investigations. PLoS ONE 11(4), 1–19 (04 2016), http://dx.doi.org/10.1371%2Fjournal.pone.0154556

4. Diamantini, C., Potena, D.: Semantic annotation and services for KDD tools sharing and reuse. In: ICDMW '08: Proceedings of the 2008 IEEE International Conference on Data Mining Workshops, pp. 761–770. IEEE Computer Society (2008)
5. Esteves, D., Moussallem, D., Neto, C.B., Soru, T., Usbeck, R., Ackermann, M., Lehmann, J.: MEX vocabulary: a lightweight interchange format for machine learning experiments. In: Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS 2015, Vienna, Austria, September 15-17, 2015. pp. 169–176 (2015), http://doi.acm.org/10.1145/2814864.2814883
6. van Harmelen, F., McGuinness, D.: OWL web ontology language overview. W3C recommendation, W3C (Feb 2004), http://www.w3.org/TR/2004/REC-owl-features-20040210/
7. Keet, C.M., Lawrynowicz, A., d'Amato, C., Kalousis, A., Nguyen, P., Palma, R., Stevens, R., Hilario, M.: The data mining optimization ontology. J. Web Sem. 32, 43–53 (2015), http://dx.doi.org/10.1016/j.websem.2015.01.001
8. Nentwig, M., Soru, T., Ngomo, A.C.N., Rahm, E.: Linklion: A link repository for the web of data. In: European Semantic Web Conference. pp. 439–443. Springer (2014)
9. Panov, P., Soldatova, L.N., Džeroski, S.: Towards an Ontology of Data Mining Investigations, pp. 257–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2009), http://dx.doi.org/10.1007/978-3-642-04747-3_21
10. Panov, P., Soldatova, L.N., Dzeroski, S.: Ontology of core data mining entities. Data Min. Knowl. Discov. 28(5-6), 1222–1265 (2014), http://dx.doi.org/10.1007/s10618-014-0363-0
11. Soru, T., Ngomo, A.C.N.: Active learning of domain-specific distances for link discovery. In: Joint International Semantic Technology Conference. pp. 97–112. Springer (2012)
12. Soru, T., Ngomo, A.C.N.: A comparison of supervised learning classifiers for link discovery. In: Proceedings of the 10th International Conference on Semantic Systems. pp. 41–44. ACM (2014)
13. Vanschoren, J., Soldatova, L.: Exposé: An ontology for data mining experiments. Proceedings of the ECML'10 International workshop on Service-oriented Knowledge Discovery pp. 31–46 (2010)
14. Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G.: Experiment databases - A new way to share, organize and learn from experiments. Machine Learning 87(2), 127–158 (2012), http://dx.doi.org/10.1007/s10994-011-5277-0
15. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: Openml: networked science in machine learning. ACM SIGKDD Explorations Newsletter 15(2), 49–60 (2014)
16. Žáková, M., Kremen, P., Železný, F., Lavrač, N.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE Transactions on Automation Science and Engineering 8(2), 253–264 (2010)
17. Widmer, G.: Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. Artificial Intelligence 146(2), 129–148 (2003)