A Hierarchical Data Mining Process Ontology

Man Tianxing, Myo Myint, Wang Guan
ITMO University
St. Petersburg, Russia
mantx626, bagyimyo1526, wagu0809@gmail.com

Nataly Zhukova
St. Petersburg State Electrotechnical University "LETI"
St. Petersburg Federal Research Center of the Russian Academy of Sciences
St. Petersburg, Russia
nazhukova@mail.ru

Nikolay Mustafin
St. Petersburg State Electrotechnical University "LETI"
St. Petersburg, Russia
ngmustafin@etu.ru

***Abstract*—Aiming to the problem of data mining processes building, we present a hierarchical data mining process ontology. The ontology is constructed based on CRISP-DM framework. Four layers of data mining process are described to solve the complexity of process synthesis. The ontology supports the management of DM processes, that assumes defining steps of the processes and selecting algorithms taking into account the input and the output data. An example of querying on the proposed ontology is presented.**

I. Introduction

Data Mining (DM) is important for a wide range of scientific and industrial domains. Since the data and requirements are diverse, there is no uniform DM process. There are lots of operators (algorithms) existing to serve the DM tasks. How should we construct a valid and applicable DM workflow based on so many available options? This issue is confusing many data researchers. Currently, they focus on the automation and overall support of DM processes [1]–[3]. The main component of the frameworks for DM processes building is the knowledge base of the DM process [4]–[7]. An effective DM knowledge base helps users to query for the current suitable solutions, these knowledge bases can be expanded with new knowledge. Ontology, as a computer-understandable language, is widely used for formal representation of objects and processes involved in scientific investigations.

In DM domain, some light-weight ontologies have been developed for the description of DM concepts [6], [8]–[11]. However, few ontologies provide a sufficient description of DM processes for the DM workflows generation. Due to the lack of description of DM processes, these semantic representations of concepts of the DM domain cannot help users build the required workflows. Although, some researches [6], [12] tried to solve this problem by building ontologies for DM processes. These ontologies describe DM processes at one level. Users have to query for a process among all the operators, that causes the complexity problem.

In this article, we present a hierarchical DM process ontology. In order to facilitate the queries, we structure the DM process description, we define four layers: phase, subprocess, action, operator. They are represented as corresponding classes and linked to each other by specific properties. The ontology is developed within the CRISP-DM framework [4]. The construction of the proposed ontology assumes reusing of the existing DM ontologies and definition of necessary concepts and relations.

The rest of this article is structured as follows: in section 2, we present the related work; then section 3 describes the design of the basic structure of the ontology; the main classes are stated in section 4; some query examples to the ontology are presented in section 5; in the last section, conclusions and future work are discussed.

II. Background

To build DM workflows effectively and correctly, it is important to have a structured framework to follow. Today some frameworks are more commonly used than others. CRISP-DM [4] is a DM framework that consists of six different phases: business understanding, data understanding, data preprocessing, modeling, evaluation, and deployment. KDD [13] is a DM framework that contains detailed descriptions of CRISP-DM phases. In KDD, more detailed phases are provided: developing and understanding of the application, creating a target dataset, data cleaning and preprocessing, data transformation, choosing a suitable data mining task, choosing a suitable data mining algorithm, employing data mining algorithm, interpreting mined patterns, and using discovered knowledge. Unlike CRISP-DM and KDD, SEMMA [14] focuses mostly on data management. The model of data mining consists of five phases: sample, explore, modify, model, and

ISSN 2305-7254

access. It doesn't consider the problem of understanding and the deployment of the generated models. At present, all the DM process frameworks are used manually to support the DM tasks. The premise of the automatic workflow generation is in the construction of a DM knowledge base.

By now the CRISP-DM framework is the most suitable framework for building DM processes, as it is more complete and has a more concise description compared to the others. There are some DM ontologies developed in recent researches. Most of them focus on the representation of DM concepts. Panov et al. proposed the OntoDM series ontologies to describe DM algorithms [8], DM process [12], and DM data type [15]. Hilario et al. summarized the characteristics of the DM algorithm while describing the concepts of the DM field to support algorithm selection [9]. However, few ontologies provide structured DM process description. The OntoKDD ontology provides abstract description of the DM process in terms of information entities and process entities, but it lacks the relations between DM processes and DM algorithms. The DMWF ontology [6] uses SWRL rules [16] to define the conditions and effects of the DM operators to support the synthesis of DM workflows. The synthesis based on DMWF is complicated. Users have to consider each condition and effect for each step. However, OntoKDD provides detailed description of abstract DM processes, while DMWF provides the description of DM algorithms. Thus, we reuse these ontologies and construct the ontology with new structure.

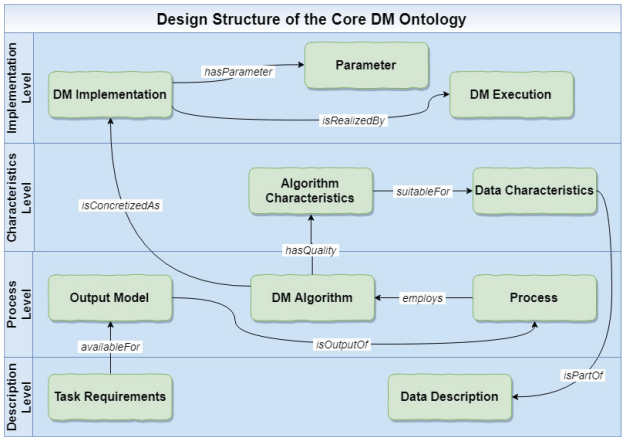

Fig. 1. The design structure of the Core DM Ontology

In our previous work, we summarized the requirements to DM tasks and constructed a multilayer Core DM ontology [11] (as Fig. 1 shows). The description of DM process is a core part of the developed DM ontology. However, using one level representation of the DM processes in practice leads to the complexity problem. In order to solve it, we propose to use hierarchical representation of the DM process [17]. Hierarchical representation allows reduce the complexity of DM workflow synthesis. Within hierarchical representation, the process is decomposed into components. Thus, in this article, we construct a hierarchical DM process ontology to support the DM workflow generation reusing the existing DM ontologies.

III. Design

The proposed DM process ontology has hierarchical structure (see Fig. 2). The layers that form the hierarchy are Phase, Subprocess, Action and Operator. The first three layers describe the abstract processes and the last one presents the processes in terms of implemented operators.

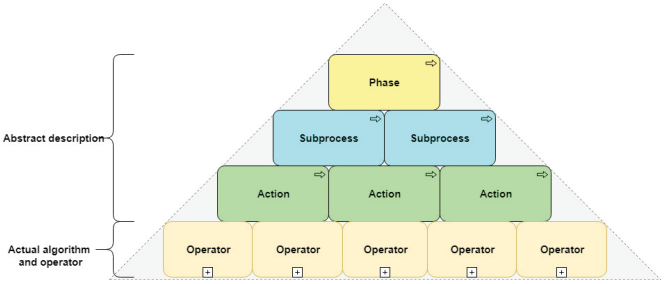

Fig. 2. The hierarchical structure of the DM process ontology

The proposed ontology assumes constructing the core hierarchical structure and linking the OntoKDD ontology [12] and DMWF ontology [6]. OntoKDD provides the abstract description of the DM processes and the DMWF provides the implemented operators and the description of the input and output objects (which are data, model, etc).

The design of the DM hierarchical process ontology is presented in Fig. 3. The "Phases" defines the general phases of DM process according to CRISP-DM framework. By the property "has_Subprocess" the "Phases" are decomposed to the "Subprocesses". For "Subprocesses", the specific input and output of each subprocess are defined. By the property "hasPostprocess", the execution order of the phases and subprocesses are defined. The "Actions" are used to describe the specific abstract actions of each subprocess in detail. Using "Operators", the realizations of the processes are described. The order of the actions and operators are defined by the restriction of their input and output.

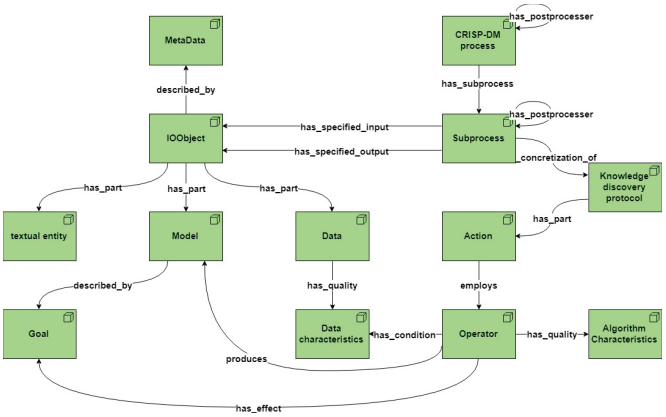

Fig. 3. The design of the hierarchical DM process ontology

In order to serve for algorithm selection, we also link the classes used to describe hierarchical processes with ”Data Characteristics” and ”Algorithm Characteristics”. The conditions and effects of the processes are defined in ”MetaData”.

The design of the proposed ontology is targeted to solve the complexity problem. If the DM processes are described at one level, the query for each process has to be performed among all operators. As the DM workflow consists of multiple steps, it is a huge work in practice. However, the proposed hierarchical structure allows query for the processes at a certain layer that limits the number of entities. Since the number of entities in specific layer is essentially less than total number of all entities, the complexity of the query for processes is significantly reduced compared to the one-level structure.Theoretical estimations of the efficiency of hierarchical processes are given in [18].

The hierarchical DM ontology is implemented based on OWLAPI interface [19] and available at *https://github.com/mantx626/Methods_For_OWL.* In this paper, we use the statistical ontology metrics from the Protégé software [20] and Bioportal [21]. The statistical metrics for the hierarchical DM ontology are shown in TABLE I.

TABLE I. THE METRICS OF THE HIERARCHICAL DM ONTOLOGY

Metrics	Value
Axiom	1902
Logical axiom count	819
Declaration axioms count	475
Class count	377
Object property count	51
Data property count	30
Individual count	4
Annotation Property count	10
Classes	377
Individuals	4
Properties	81
Maximum depth	6
Maximum number of children	27
Average number of children	3
Classes with a single child	18
Classes with more than 25 children	1
Classes with no definition	375

IV. MAIN CLASSES

In constructing the proposed ontology, we defined the following key classes that contain entities for describing processes at different levels:

The “Phases” class describes an iterative process based on CRISP-DM framework, that starts with getting a business understanding of the task. An understanding of the data is obtained and then, the data is processed for the further modeling step where the actual DM algorithms are defined. The result produced by the output model is then evaluated. If the result meets the requirements to accuracy, speed, etc, the algorithm is being deployed. Targeted for general DM process description, this layer provides the main categories for process query and defines basic execution order of the processes by the property ”hasPostprocess” (see Fig. 4). Users can focus on one or more phases and then dig into more details.

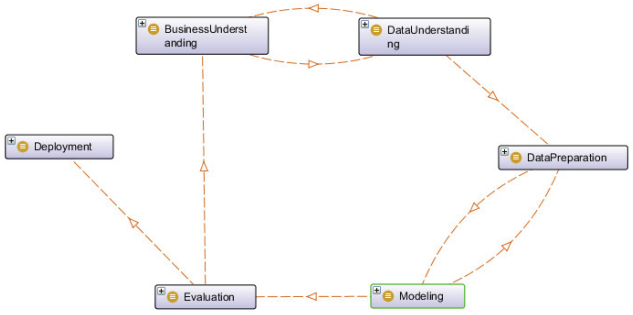

Fig. 4. The ”Phases” class in the hierarchical DM ontology (The arrows mean the property ”hasPostprocess”)

The ”Subprocess” class describes the generic entities from CRISP-DM framework (see Fig. 5). We represent them as subprocesses with specific input and output.This class completely and stably covers the entire DM process. For example, the ”Data Preparation” phase consists of five subprocesses: ”CleanData”, ”FormatData”, ”SelectData”, ”ConstructData”, and ”IntegrateData”.

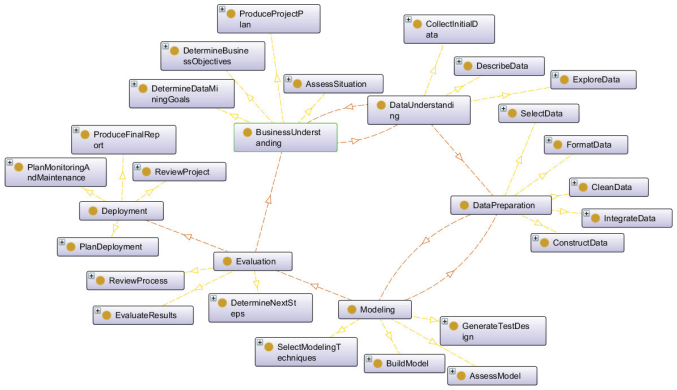

Fig. 5. The ”Subprocess” class in the hierarchical DM ontology (The brown arrows mean the property ”hasPostprocess” and the yellow arrows mean the property ”has_Subprocess”)

The ”Action” class provides all possible actions on the data (see Fig. 6). This class is implemented by reusing the OntoKDD ontology [12]. The general categories of the actions include: acquisition, assess action, identify action, and so on. In each category, there are multiple specific actions. An example of the specific actions in the class ”Assess action” is shown in Fig. 7. The detailed realization of each subprocess in terms of actions is specified by class ”Knowledge discovery protocol”(see Fig. 3).

The ”Operator” class specifies the implemented operators and actual DM algorithms. Unlike the abstract description of the process in terms of phases and subprocesses, this class provides the specific description ”what we should do”

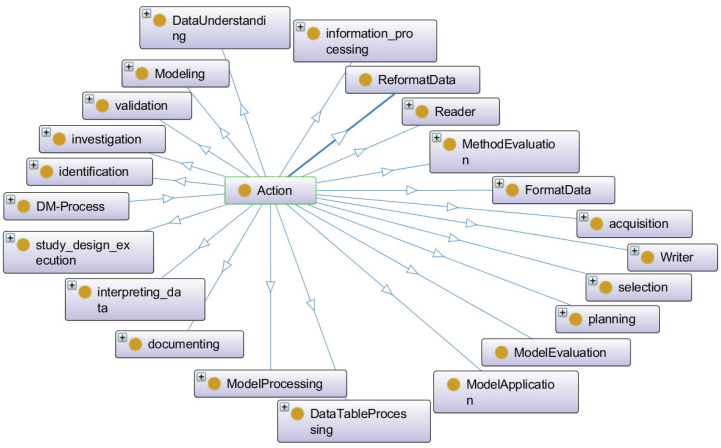

Fig. 6. The main subclasses of the class ”Action” in the hierarchical DM ontology(The blue arrows mean the property ”has_Subclass”)

from the implementation perspective. There are two types of operators. The first type of operators are implemented by basic operators, such as ”Reader”, ”Writer”, and so on (see Fig. 8). The implementation of the second type of operators employs the DM algorithms. We reused the ”Modeling” class from DMWF ontology [6]. It extracts the learner operators from the software ”RapidMiner” [22]. Thus, these operators have the prefix ”RM”. An example of the operator ”Clusterer” (which is a subclass of ”UnsupervisedLearner” in Fig. 8) in the proposed ontology is shown in Fig. 9.

Except the four classes for describing the DM process mentioned above, some additional classes are defined to support the description of the DM process:

The class “IOObjects” presents the objects that are defined or produced by the processes. The upper subclasses of ”IObjects” (”Textual entity”, ”Data”, ”Model”) are presented in Fig. 10. Each IOObject produced by a process can be used as an input of any process occurring at some later stage of the DM process. For example, ”Model” can be produced as the output by ”RM_ID3”, while used as the input by ”ModelPruning”.

The class ”MetaData” is used to describe the “IOObjects” in detail. The upper subclasses of “MetaData” (”AttributeType”, ”DataColumn”, ”DataFormat”, ”Attribute”) are presented in Fig. 11.

Some other related classes are imported from Core DM ontology [11], such as “Characteristics” which describes the characteristics of data and algorithms; “Goal” which describes the task requirements. They provide a basis for algorithm selection.

V. CASE STUDY

The hierarchical ontology serves to support the query for DM processes at different layers. It has low computational complexity, so it is able to handle the complicated processes (see Fig. 12). In order to construct the hierarchical structure, two components are defined for each process:

1) I/O Objects are used to describe the input and output of each process. The properties ”has_specific_input” and ”has_specific_output” link DM process entities and the entities of the class ”IOObjects”. Since the input of processes are also the output of another process, the I/O objects can be used as the conditions for ordering the processes.
2) Decomposition assumes defining subprocesses of each process and thus form a hierarchical representation of the process. These decomposition can help users to explore more details about a process by considering specific properties(”has_subprocess”, ”is_concretizationof”, ”employs”).

Therefore, when the users query the proposed ontology to get DM process, a general process is provided firstly. Then, the

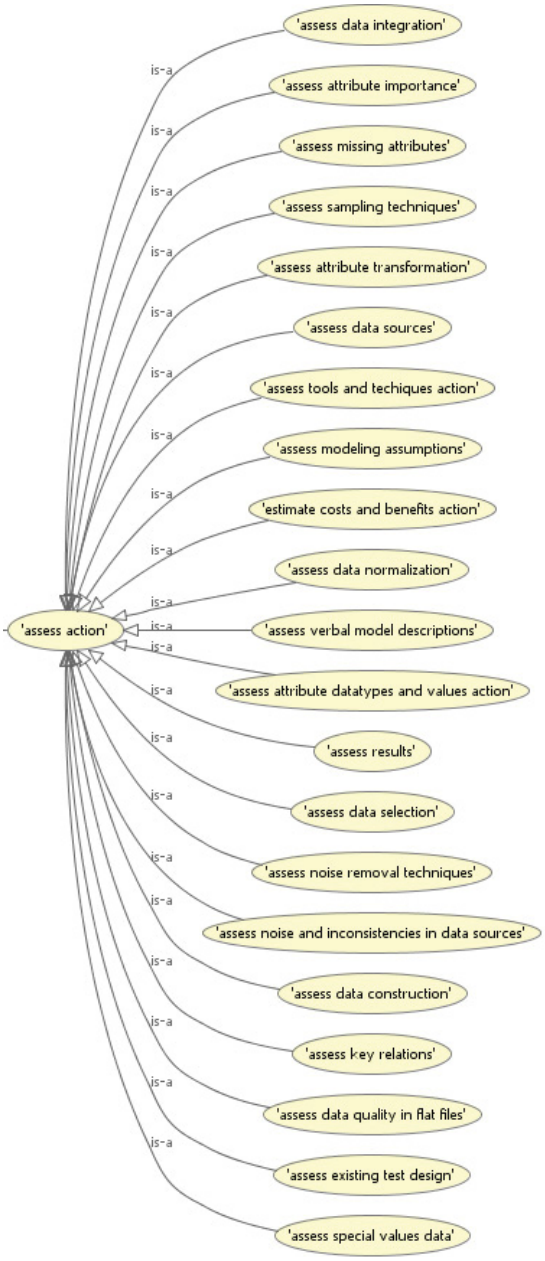

Fig. 7. The specific actions in class ”Assess action”

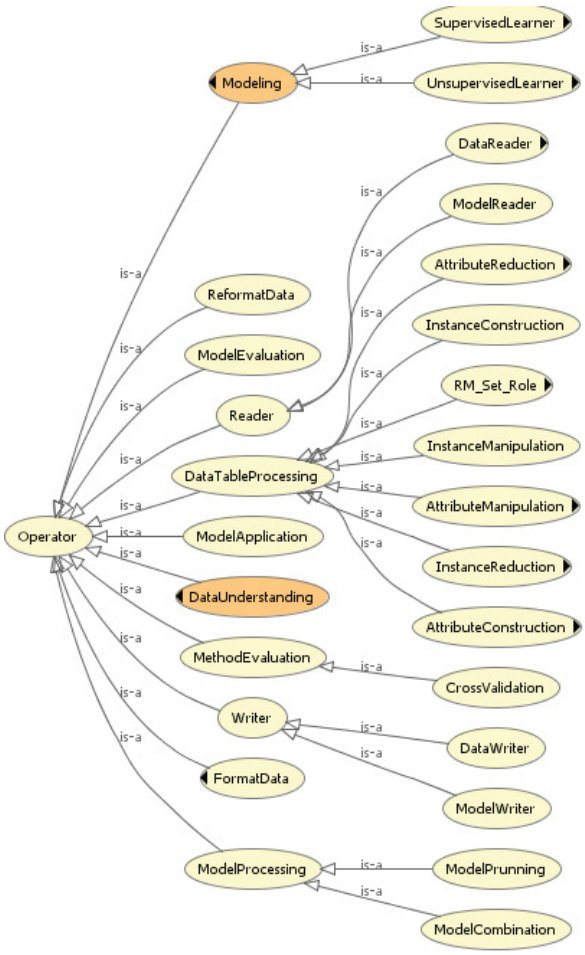

Fig. 8. The basic operators in class ”Operator”

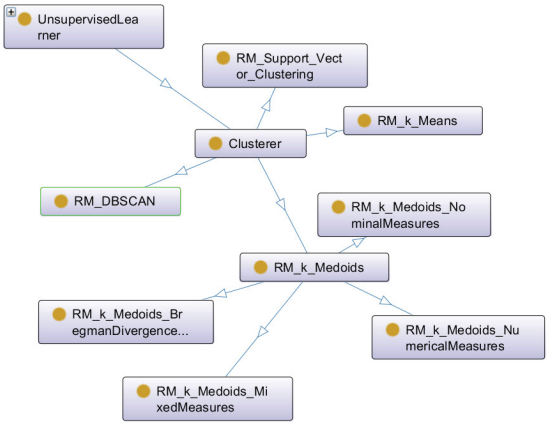

Fig. 9. The operator ”Clusterer” in the hierarchical DM ontology

queries are performed to lower layers of the hierarchy based on decomposition. With the conditions provided by the I/O objects, a sequence of the process is also obtained.

As an example, a simplified DM process query is presented in Fig. 13.

In the beginning, we obtain the general DM process with

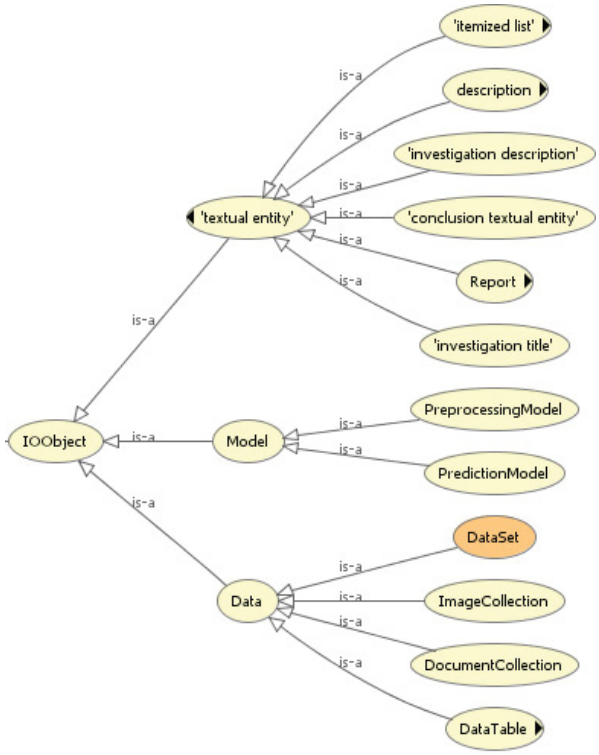

Fig. 10. The upper subclasses in class ”IOObjects”

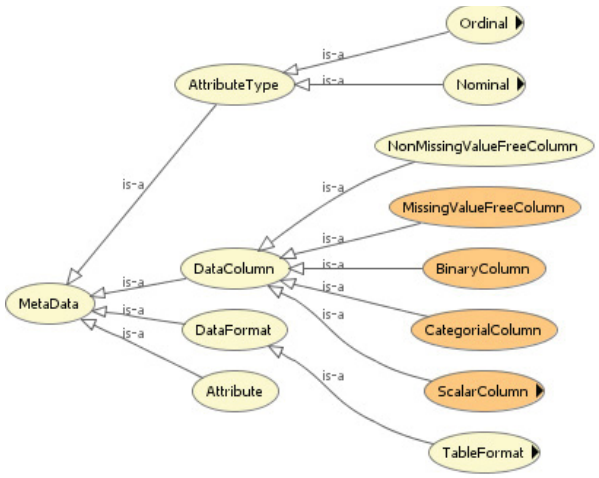

Fig. 11. The upper subclasses in class ”MetaData”

the input ”data” and the output ”model”. Then, we focus on the phase ”DataPreparation”, which input and output are both ”data”. According to the property ”has_subprocess”, we can explore the subprocess ”ConstructData”, which assumes several actions: ”data normalization”, ”handling missing attribute values”, and so on. At the layer that defines actions of the processes, more details about the actions are provided. For example, the action ”handling missing attribute values” has subactions: ”imputation”, ”delete”, and ”ignore”. As the output of ”imputation”, the missing values are filled. The implemented operators are obtained according to the property ”employs”.

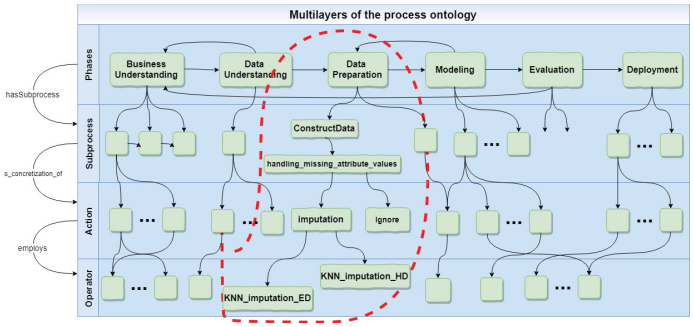

Fig. 12. Query the DM process at different layers of the hierarchical DM ontology

The available operators of the action ”imputation” include ”KNN_imputation_ED” (KNN algorithm employing euclidean distance), ”KNN_imputation_HD” (KNN algorithm employing hamming distance), ”Filling_manually” and so on. The I/O objects serve for ordering the process and algorithm selection. As Fig. 13 shows, the input data is continuous and has missing values not random. Thus, the ”KNN_imputation_ED” (KNN algorithm employing euclidean distance) is suitable for this situation. Since the output is the data without missing values, the imputation can be placed before modeling algorithms which have input ”data *and* ***hasno*** *some* missing_values”.

```
Main Task: DM process
    I/O:
        has_specific_input: data
        has_specific_output: model
    Decomposition:
        has_a:
            BusinessUnderstanding
            DataUnderstanding
            DataPreparation
            …
            Deployment
        …
"Phase" layer Task: DataPreparation
    I/O:
        has_specific_input: data
        has_specific_output: data
    Decomposition:
        has_subprocess:
            CleanData
            ConstructData
            …
            IntegrateData
        …
"Subprocess" layer Task: ConstructData
    I/O:
        has_specific_input: data
        has_specific_output: data
    Decomposition:
        is_concretization_of data construction protocol has_part:
            data normalization
            handling missing attribute values has_a:
                imputation
                delete
                ignore
            …
            assess data construction
        …
"Action" layer Task: imputation
    I/O:
        has_specific_input: DataColumn and has some missing values
        has_specific_output: data and hasno some missing values
    Decomposition:
        employs:
            KNN_imputation_ED
            KNN_imputation_HD
            …
            Filling_manually
        …
"Operator" layer Task: KNN_imputation_ED
    I/O:
        has_specific_input: DataColumn and has some (missing values
and hastype only Missing_Not_Random) and hastype only continuous
        has_specific_output: DataColumn and hasno some missing
values
```

Fig. 13. An example of representation of the hierarchical DM process based on the hierarchical DM ontology

VI. Conclusion

In this article, we present a hierarchical DM process ontology. The construction assumes reusing existing DM ontologies (OntoKDD [12], DMWF [6]). Unlike the traditional description, we construct the processes descriptions with a hierarchical structure to solve the complexity problem of process synthesis. Four layers are defined: ”Phases”, ”Subprocess”, ”Action”, and ”Operator”. On the first three layers the description of the processes is abstract, and the last layer is the description of the processes implementation. We also present the representation of input and output, which define the restrictions serving to order the DM processes, and the representation of characteristics of data and algorithms, which support the algorithm selection.

A simplified example of querying on the proposed ontology is presented. Users can obtain the DM process with different level of detail. In the future work, we will focus on linking the proposed ontology with the existing DM ontologies, and then implement the automatic generation of DM workflows.

References

[1] Q. Yang and X. Wu, “10 challenging problems in data mining research,” *International Journal of Information Technology & Decision Making*, vol. 5, no. 04, pp. 597–604, 2006.

[2] H. Kriegel *et al.*, “Future trends in data mining, data mining and knowledge discovery,” *15: 87*, vol. 97, 2007.

[3] T. G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, and P. Tadepalli, “Structured machine learning: the next ten years,” *Machine Learning*, vol. 73, no. 1, p. 3, 2008.

[4] R. Wirth and J. Hipp, “Crisp-dm: Towards a standard process model for data mining,” in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. Springer-Verlag London, UK, 2000, pp. 29–39.

[5] M. Žáková, P. Křemen, F. Železný, and N. Lavrač, “Automating knowledge discovery workflow composition through ontology-based planning,” *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 253–264, 2010.

[6] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, “Towards cooperative planning of data mining workflows,” 2009.

[7] M. Hilario, A. Kalousis, P. Nguyen, and A. Woznica, “A data mining ontology for algorithm selection and meta-mining,” in *Proceedings of the ECML/PKDD09 Workshop on 3rd generation Data Mining (SoKD-09)*, 2009, pp. 76–87.

[8] P. Panov, S. Džeroski, and L. Soldatova, “Ontodm: An ontology of data mining,” in *2008 IEEE International Conference on Data Mining Workshops*. IEEE, 2008, pp. 752–760.

[9] C. M. Keet, A. Ławrynowicz, C. d’Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, and M. Hilario, “The data mining optimization ontology,” *Journal of web semantics*, vol. 32, pp. 43–53, 2015.

[10] K. Benali and S. A. Rahal, “Ontodta: Ontology-guided decision tree assistance,” *Journal of Information & Knowledge Management*, vol. 16, no. 03, p. 1750031, 2017.

[11] M. Tianxing, A. Vodyaho, N. Zhukova, N. Mustafin, and A. M. Thaw, “Meta mining ontology framework for domain data processing,” in *Conference of Open Innovations Association, FRUCT*, no. 26. FRUCT Oy, 2020, pp. 667–674.

[12] P. Panov, L. Soldatova, and S. Džeroski, “Ontodm-kdd: ontology for representing the knowledge discovery process,” in *International Conference on Discovery Science*. Springer, 2013, pp. 126–140.

[13] R. J. Brachman and T. Anand, “The process of knowledge discovery in databases: A first sketch.” in *KDD workshop*, vol. 3, 1994, pp. 1–12.

[14] U. Shafique and H. Qaiser, “A comparative study of data mining process models (kdd, crisp-dm and semma),” *International Journal of Innovation and Scientific Research*, vol. 12, no. 1, pp. 217–222, 2014.

[15] P. Panov, L. N. Soldatova, and S. Džeroski, “Generic ontology of datatypes,” *Information Sciences*, vol. 329, pp. 900–920, 2016.

[16] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean *et al.*, "Swrl: A semantic web rule language combining owl and ruleml," *W3C Member submission*, vol. 21, no. 79, pp. 1–31, 2004.

[17] D. Draheim, V. Geist, and C. Natschlaeger, "Integrated framework for seamless modeling of business and technical aspects in process-oriented enterprise applications," *International Journal of Software Engineering and Knowledge Engineering*, vol. 22, no. 05, pp. 645–674, 2012.

[18] V. Y. Osipov, A. I. Vodyaho, N. A. Zhukova, and P. A. Glebovsky, "Multilevel automatic synthesis of behavioral programs for smart devices," in *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*. IEEE, 2017, pp. 335–340.

[19] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," *Semantic web*, vol. 2, no. 1, pp. 11–21, 2011.

[20] M. A. Musen, "The protégé project: a look back and a look forward," *AI matters*, vol. 1, no. 4, pp. 4–12, 2015.

[21] N. F. Noy, N. H. Shah, P. L. Whetzel, B. Dai, M. Dorf, N. Griffith, C. Jonquet, D. L. Rubin, M.-A. Storey, C. G. Chute *et al.*, "Bioportal: ontologies and integrated data resources at the click of a mouse," *Nucleic acids research*, vol. 37, no. suppl_2, pp. W170–W173, 2009.

[22] M. Hofmann and R. Klinkenberg, *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, 2016.