Design Patterns for a Knowledge-Driven Analytical Platform Core

Viktor Zayakin
Business Informatics Department
HSE University
38 Studencheskaya str., Perm, 614070,
Russian Federation
vszayakin@yandex.ru

Lyudmila Lyadova
Business Informatics Department
HSE University
38 Studencheskaya str., Perm, 614070,
Russian Federation
lnlyadova@gmail.com

Evgeny Rabchevsky
SEUSLAB LLC
111 Shosse Kosmonavtov, Perm,
614066, Russian Federation
e.rabchevskiy@seuslab.ru

***Abstract* — The development and support of knowledge-based systems for experts in the field of social network analysis (SNA) is complicated because of the problems of viability maintenance that inevitably emerge in data intensive domains. Largely this is the case due to the properties of semi-structured objects and processes that are analyzed by data specialists using data mining techniques and others automated analytical tools. In order to be viable a modern knowledge-based analytical platform should be able to integrate heterogeneous information, present it to users in an understandable way and to support tools for functionality extensibility. In this paper we introduce an ontological approach to information integration and propose design patterns for developing analytical platform core functionality such as ontology repository management, domain-specific languages (DSLs) generation and source code round-trip synchronization with DSL-models.**

***Keywords* — information integration, knowledge bases, databases, system viability, analytical platforms, open data, data analysis**

I. Introduction

The development of knowledge-driven analytical systems in data-intensive domains (e. g. social network analysis) is inevitably accompanied by problems of maintaining viability of those systems. Such class of intelligent systems shares common properties of development and support processes that result in common design challenges.

Firstly, logical models of knowledge bases (KB) and databases (DB), used by intelligent system for modeling the analyzed objects and processes, as well as to describe the ways of their physical representation, are constantly expanding and being modified, which creates *problems of information integration*. To a large extent, this is due to the properties of semi-structured objects and processes being analyzed, as well as the intensive use (gathering, processing, generation, etc.) of data.

Secondly, a wide range of specialists are involved in the development of intelligent system such as domain experts, knowledge engineers, data analysts, software engineers, etc. Each of these groups of specialists may interpret the integrated information differently, depending on the domain context in which they operate. Thus, it is crucial for analytical platform to successfully solve *problems of interpreting same data according to different data and domain models*.

Thirdly, the analytical platforms that are used in data-intensive domains are required to be extensible and allow users to implement or specify algorithms for data processing (including the use of data mining and machine learning techniques) themselves. Considering that the majority of end users (in particular, domain experts) do not have programming skills needed to implement such functionality, this creates *problems of platform's extensibility* and requires providing users with domain-specific modeling tools.

For developing a consistent and comprehensive solution to the mentioned groups of problems in this paper we identify common design challenges that arise in the process of development of knowledge-driven analytical platforms in data-intensive domains. As a result, this research proposes a conceptual ontological approach to information integration based on rules as well as design patterns for knowledge-driven analytical platform core functionality based on OWL 2 models of its abstract syntax.

II. Requirements and Design Challenges

Tools for development of viable knowledge-driven systems should be able primarily to integrate heterogeneous information, support changes traceability in data and knowledge models and interpret information according to the domain model that is understandable to the data consumer [1]. However, the development of an analytical platform is not feasible for an ordinary developer, since it involves the development of methods for solving specialized problems and the design of domain models that could include fuzzy relationships, fuzzy terms, temporal and spatial knowledge. In this regard, the solution of such a complex problem should be focused primarily on the creation of core tools, invariant to the domains in which they are used and that could be used to implement base functionality of a platform [2].

The following common requirements can be applied to the development tools of knowledge-driven analytical platforms in data-intensive domains:

1. Ensuring extensibility of logical models when descriptions of new sources of the information, such as data sources and domain models are included. It also requires automatic checks of logical integrity of the integrated model.
2. Ensuring integration of subject data with results of their analysis when algorithms of automated data processing (e. g. data mining techniques) are applied.
3. Providing tools for independent interpretation of data and results of their processing according to different domain models.
4. Ensuring traceability of changes in metamodels to support the relevance of semantic annotation of stored information.
5. Providing tools for declarative specification of data processing as well as input/output data structures used to extract/write data from/into DBs before/after executing data processing algorithms.
6. Providing tools for integration of data and knowledge

models with software modules which implement data processing algorithms.

Related work in the field of knowledge-driven analytical platforms development includes the development of approaches to information integration and data federation. The vast majority of existing approaches are based on ontologies [3]. These approaches combine ontological descriptions of integrated sources and the descriptions of mappings and transformations between ontologies and information sources into a multifaceted ontology [4, 5].

Existing approaches are specifically designed to independently model and integrate various aspects of an intelligent system, namely, data structures (databases, event logs, texts), domains, data processing tasks, domain-specific languages, etc. This approach also provides a straightforward extension of the integration model, allowing to embed a new ontology by describing axiomatic relations between its elements and elements of previously formed ontologies. The implementations of multifaceted ontology-based information integration are used to design data architecture for networked enterprises [6], integrate spatial DBs [7], construct or transform queries to distributed DBs [8, 9, 10], describe and form datasets for machine leaning tasks [11].

Based on the analysis of modern tools for the development of knowledge-driven analytical platforms, following common design challenges could be stated:

1. Integration of information (data sources models, data processing problems descriptions, domain models) at the level of source metamodels which are described independently of each other. It allows for metamodels to be described by different teams or to be imported from the Web and reused in the process of system development and maintenance.
2. Automated interpretation of data based on formal description of logical constraints and domain rules. It allows to use single inference subsystem to restructure data according to different models which helps to deduplicate and/or reduce the volume of stored information.
3. DSL integration for specification of data processing modules for gathering, preprocessing, analyzing and interpreting data. It requires the development of DSM subsystem. In exchange this provides users with tools to extend the functionality of the platform independently of its developers.
4. Integration of problem solvers based on declarative specifications of platform modules. It allows to generate specified versions of scripts and applications and to call them by providing a declarative description of input data, and parametrized attributes.
5. Composition of data processing modules into pipelines based on input and output data structures descriptions. It allows to automatically infer in which sequence data processing modules can be called. This can be used to orchestrate data processing tasks and manage data flows by such systems as Apache NiFi.

III. Ontological Approach to Information Integration

The problem of information integration appears to be central in knowledge-driven analytical platform development. Without solving this problem, it is problematic to provide users with data to analyze in a transparent and comprehensible way which creates difficulties for them when choosing analytical techniques, methods and tools for data to be processed. Considering that in data-intensive domains data models are iteratively evolving it is crucial to develop an approach to information integration which would serve as a framework for managing multiple sources description.

Though existing approaches [6-11] allow to model and integrate source descriptions, they are restricted in terms of managing different versions of the same models resulting in changes traceability difficulties. For this reason, we introduce an ontological approach to information integration which helps to overcome described problem.

The proposed approach to information integration helps to organize information that is used in the processes of applying methods of automated data analysis and interpreting the results of analysis using expert knowledge of domain (Fig. 1).

The main idea of the approach is to model various information sources (using relevant terminology for context of the source) independently using ontologies. Then, elements of integrated ontologies are linked (or mapped) together using production rules or logical restrictions which allows automatic inference using semantic reasoners (e. g. Pellet, ELK, HermiT, etc.). There is a set of languages that could be used to formally describe ontologies and rules such as OWL 2, SWRL and RIF.

This approach can be called interpretive. Initially, ontology individuals are annotated according to the concepts in the ontology that describes some data source. When used to solve analytical problem, individuals are interpreted in terms of an ontology that models the conceptual scheme of this problem, according to the rules or logical restrictions. Similarly, the results of the analysis which are annotated using the concepts of the problem ontology, then are interpreted in the context of the ontology that models domain defined by experts and conceptualized by knowledge engineer.

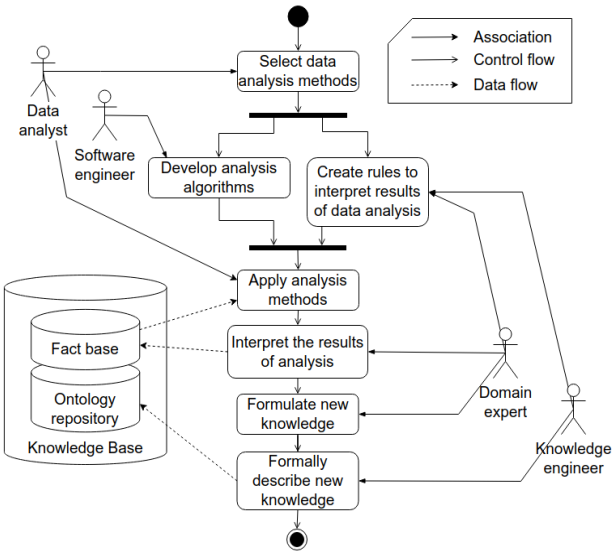

Fig. 1. The model of process of data analysis and interpretation

The process of data analysis and interpretation includes following steps:

1. A data analyst, based on the analytical problem being solved, determines the method for solving it.
2. A software engineer implements algorithms for selected method(s).
3. A domain expert and a knowledge engineer formulate

rules and logical restrictions for interpreting the results of data analysis based on the possible outputs of the analysis methods.

4. A domain expert formulates new knowledge about the domain after interpreting the results.

5. Corresponding knowledge models then can be refined by a knowledge engineer by describing new patterns based on the extracted knowledge.

Knowledge base (Fig. 2) is structurally divided into three groups of ontologies according to the type of modeled information:

1. Knowledge about data and data sources (Data Source ontologies). They may include information about data types, data storage formats, relations arity, attributes, etc.

2. Knowledge about data processing problems that use subject data to infer new facts and extract new knowledge (Problem ontologies). They may include information about the structure of input and output data for data processing procedures (e. g. data mining algorithms), links to externally executed scripts, the sequence of procedure calls, etc.

3. Domain knowledge (Domain ontologies). They may include concepts of the domain, as well as axiomatic statements that model the limitations of the domain.

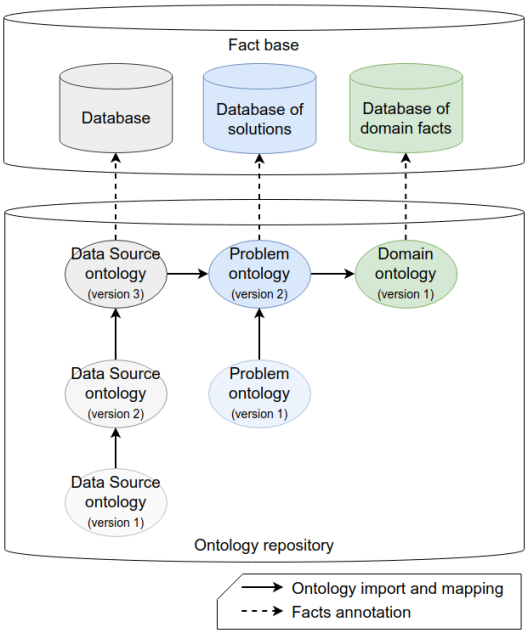

Fig. 2. The basic structure of knowledge base

In practice, the proposed approach allows to avoid data duplication. Data interpretation is determined while processing user query according to the context defined by ontologies. At the same time, it is possible to trace which facts and at what stage of processing was fetched or inferred. Facts gathered from a source (for example, from a specific social network) would be annotated using an ontology that describes this source. The results of data analysis (for example, the identifiers of duplicated objects) would be annotated using concrete problem ontology. This increases the transparency of the data and allows to present the same information to different platform users according to the terminology they are familiar with.

Moreover, the proposed approach allows to manage and integrate different versions of ontologies to ensure traceability of changes in the same fashion as different types of ontologies which is not supported by existing approaches. We can simply copy an old version of ontology, modify it and define new rules and logical restrictions to bind elements of two versions of the same ontology.

IV. Description of Design Patterns

A. An Approach to Analytical Platform Development

According to [13], in order to develop a modern knowledge-driven analytical platform it is crucial to define its core functionality and to create tools which will allow to implement its base modules for gathering, preprocessing, analyzing and interpreting data. Then it will be possible to extend platform specifying new functionality by adapting its base modules.

In this paper we propose (Fig. 3) that new modules of an analytical platform are specified by users using a set of visual domain-specific languages generated on top of ontologies which are integrated in the knowledge base [5]. As a result, using generated DSLs, users who are not familiar with programming will be able to specify the modules and functions of the platform adapting existing functionality to their specialized problems [14]. At the same time, to support the uniformity of the resources description, DSLs themselves can also be represented as ontologies.

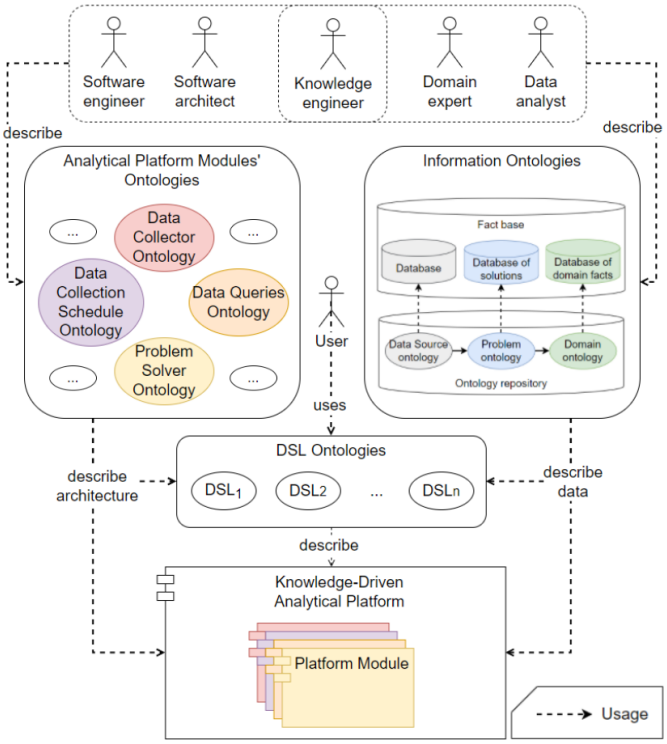

Fig. 3. An Approach to Analytical Platform Development

An approach to information integration, in that regard, is used to describe data structures of inputs and outputs that could be queried from fact base and passed into platform modules instances. Besides, analytical platform modules' ontologies form base functionality of concrete analytical platform (for instance, platform for social network analysis) to be implemented using core functionality of a platform and could import, reuse and extend integrated information ontologies as well.

In order to execute approach to analytical platform development core functionality should allow to:

- manage ontologies;

- generate DSLs and visual editors for them;
- synchronize DSL-models with source code of the platform modules;
- execute problem solvers;
- manage problem solvers pipelines;
- etc.

Thus, it is important to develop design patterns for the common core of different analytical platforms which could be developed using this approach. The specification of OWL 2 language [12] is used for this purpose. It contains the set of UML class diagrams and allows to extend them introducing new elements (which are highlighted in **bold**) to form the patterns.

B. *Pattern for Importing Ontologies*

The pattern for importing ontologies (Fig. 4) describes the taxonomy of integrated sources descriptions, as well as the types of logical restrictions that can be used to link elements of several ontologies.

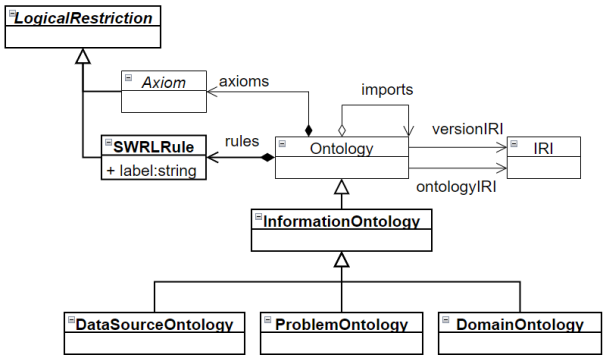

Fig. 4. The model of the pattern for integrating ontologies

Ontology which describes an integrated source of information is modeled by *InformationOntology* class which has three subclasses (*DataSourceOntology*, *ProblemOntology*, *DomainOntology*) according to every type of ontologies in the knowledge base. In order to identify ontology and all its versions IRIs are used.

Elements of ontologies (classes, object properties, datatype properties, etc.) are linked using either *SWRLRules* or *Axioms* that could be modeled using OWL 2 expressive capabilities. *SWRLRules* are one of the ways to define ontological mappings and logical restrictions within ontology. Just like axioms, they can be interpreted by reasoners (e.g., Pellet) to obtain new facts based on the information that gathered from data sources and stored in the fact base.

C. *Pattern for Ontology-Based Metamodeling*

The pattern for ontology-based metamoding (Fig. 5) is intended to be able to create metamodels are defined in the form of ontologies for describing DSLs and input/output data structures for problem solvers on top of integrated in knowledge base ontologies.

For each *Ontology* could be defined *SubOntology* that specify it. It is used to choose elements of the specified ontology that will form a model of an integrated DSL. It is constrained that *SubOntology* only can specify existing elements on an ontology, which is why elements added into *SubOntology* can only be *SubEntities* (e. g. subclasses and subproperties).

AssertionSubOntology which is the subtype of *SubOntology* is also used to specify ontologies but only by adding individuals and assertions involving them. It allows to ensure that new elements and axioms in *AssertionSubOntology* do not change the structure of metamodel that is defined by specified ontology. *AssertionSubOntologies* are supposed to be used to define concrete models according to the DSL and specify data structures of inputs and outputs for platform modules.

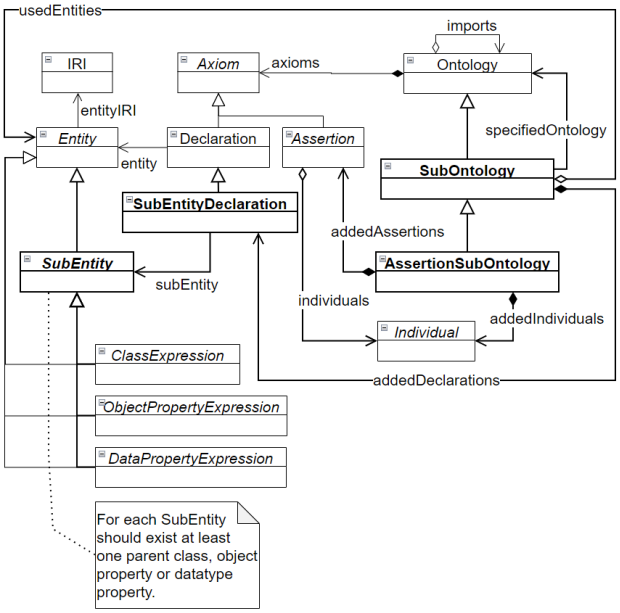

Fig. 5. The model of the pattern for ontology-based metamodeling

D. *Pattern for Integrating DSLs*

The pattern for integrating DSLs (Fig. 6) specify the ontological metamodeling pattern for generating and integrating DSLs to create languages to specify data processing modules in order to formally describe and manage the architecture of the analytical platform.

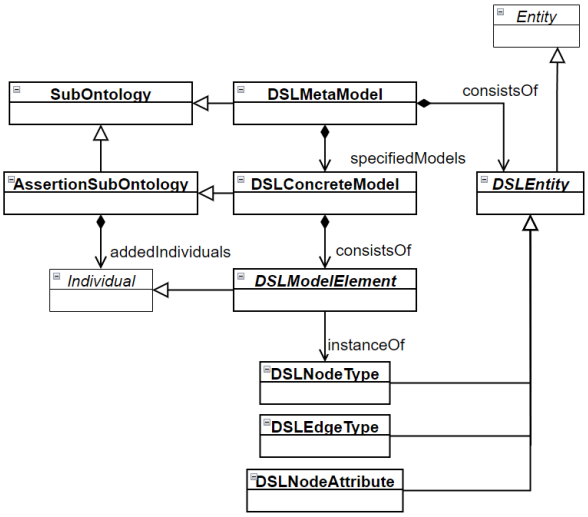

Fig. 6. The model of the pattern for integrating DSLs

Classes and properties of ontologies that are specified by *SubOntology* can used to form *DSLMetaModel*, and *DSLConcreteModel* is used to define models using DSL. *DSLConcreteModel* is a subclass of *AssertionSubOntology*, thus, added elements of DSL-models are individuals. The definition of *DSLEnities* as ontology classes and properties

allows to integrate them for inference of new facts which could be then automatically conceptualized in terms of *InformationOntologies*.

E. Pattern for Integrating Problem Solvers

The pattern for integrating problem solvers (Fig. 7) describes the structure of entities that are used to implement data processing modules of a platform using generated DSLs. The implementation of this pattern requires to coordinate the processes of designing platform modules using DSL, implementing algorithms for problem solvers, and modeling the structure of input and output data for the problems to be solved by platform modules.

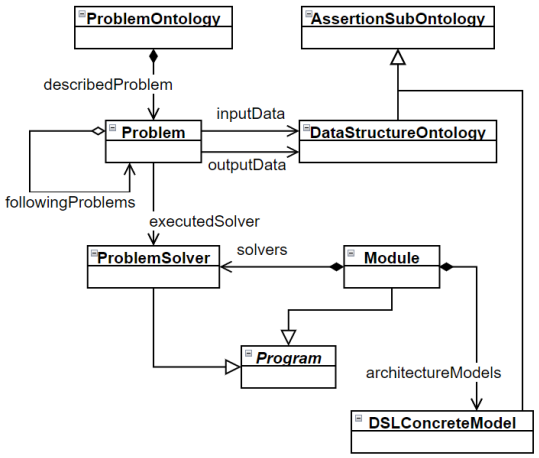

Fig. 7. The model of the pattern for integrating problem solvers

A set of *DSLConcreteModels* is used to describe an architecture of a platform's Modules which are interpreted by platform's core. Every *Module* includes *ProblemSolvers* that are executed by the core. Each *ProblemSolver* are linked to the *Problem* description which is defined by some *ProblemOntology*. *DataStructureOntology* specifies input and output data structures of a *Problem* which allows to infer if one *Problem* can be solved *following* another one when defining a data processing pipeline.

V. Functionality of Analytical Platform Core

It is assumed that the developed design patterns can be used to develop the analytical platform core. In order to demonstrate the principal possibility of creating such a subsystem, in this section we describe the functionality of the core.

We propose that the knowledge engineer is responsible for managing the ontology repository (Fig. 8). He can *create ontologies* in a form of files corresponding to the OWL 2 with the assignment of a URI for the ontology and all its elements (classes, properties, etc.). Once created, the ontology file should be available for opening externally in an ontology editor (e.g., Protege) to directly fill the ontology with axioms and rules.

It is also possible to create the special kinds of ontologies based on the existing ones. As was mentioned, a subontology (*Create a Subontology*) allows definition of a new element (class, property) if it is located within the existing taxonomies of an original ontology. In an assertion subontology (*Create an Assertion Subontology*) it is strictly individuals that are available for creation, as well as the assignment of property values to them.

In the process of creating a subontology, the knowledge engineer selects a part of the taxonomies of classes and properties that are transferred to the subontology. This requires performing a reasoning to restore domains and ranges of properties if they are represented by classes that are removed from the taxonomy of the original ontology.

In addition, it is crucial to be able to *Validate a subontology* to verify the consistency of the creation of any elements in a subontology (or an assertion subontology) within the taxonomies of the original ontology.

If it is needed to use an existing ontology it can be imported into the repository (*Import ontology*) from a remote server or a local file. After the import, a duplicate ontology is created separately within repository not to cause any conflict with files in the internal file system of the host operating system. For the purpose of building ontologies using existing ones there is a use case *Reference an ontology*, which allows elements of an existing ontology to be reused in the selected ontology.

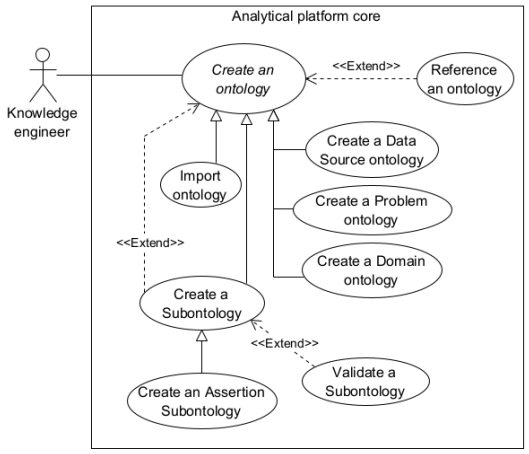

Fig. 8. The functionality of managing the repository of ontologies

According to an approach to analytical platform development, the software architect can use the platform core to *create DSLs* (Fig. 9) which provides users with tools to specify platform modules in a declarative way using visual editors. Using the pattern for integrating DSLs we can easily define which elements of selected ontologies (i. e. which parts of taxonomies) form a concrete DSL metamodel using the level of abstraction that is provided by subontologies.

It is proposed that classes of the ontology that describe a DSL are used to represent classes of model elements (node types), object properties are used to depict relationships between classes (edge types) and datatype properties are used to list classes' attributes. Thus, a concrete model that is created according to DSL can only contain elements, edges between them and attribute values that can be represented as an assertion subontology.

In addition, the core allows the software engineer to *generate source code* of platform modules. For instance, we can transform an ontology as a set of classes that are translated to Python 3 programming language to be used as a blueprint of a problem solver.

Similar to DSL metamodels, the data engineer (Fig. 10) can define metamodels of input and output data structure (*Describe data structure for a problem*) as a subontologies. Then, based on them the data analyst can describe and extract

a dataset that is passed into a problem solver when the problem is being solved. Also, he has the possibility to validate the dataset according to the metamodel that defines its structure. When the data processing task is completed, validation can also be applied to the result of data processing.

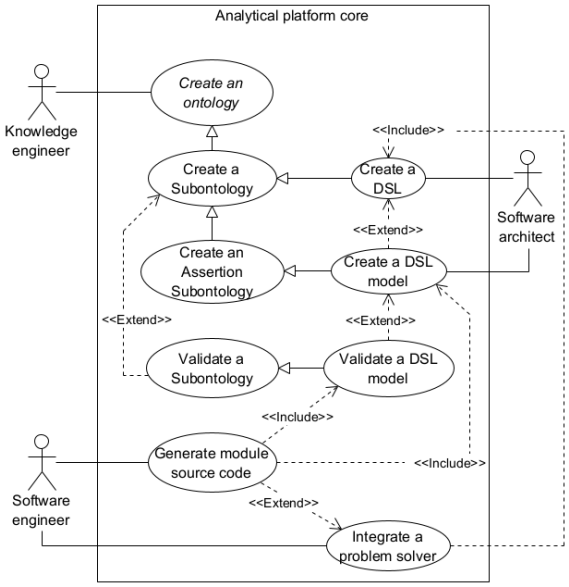

Fig. 9. The functionality of integrating DSLs and problem solvers

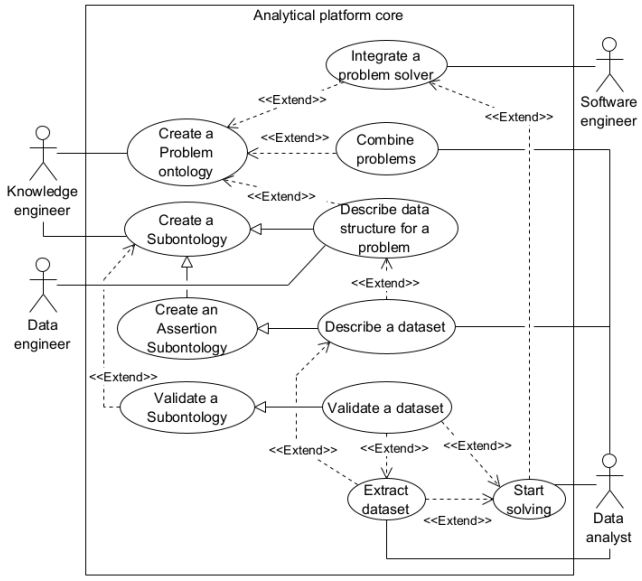

Fig. 10. The functionality of managing problem solving

Using the core users (namely, data analysts) can extend the platform functionality by simply specifying DSL models of a new module that is relied on a problem solver. This helps to reduce their dependence on software engineers and increase an adaptivity of a platform resulting in its improved viability.

VI. Conclusions and Future Work

In this paper we proposed an ontological approach to information integration that allows to integrate different types of ontologies which are relevant to data-intensive domains (data source metamodels, problem descriptions, domain models). An interpretive approach to integration helps to avoid data duplication, ensure changes traceability of ontologies and automatically interpret data and the results of data analysis to provide them to different groups of users according to terminology that they are familiar with.

In order to create a basis for solving the design challenges of developing knowledge-based analytical platforms we presented design patterns based on the models of OWL 2 language. It is expected that the developed models will be used to implement analytical platform core functionality that will help to provide users with the tools to extend platform functionality with minimal reliance on software engineers.

Further work in this field is aimed at implementing software prototypes of the platform's core functionality based on described patterns and ontological approach to information integration.

References

[1] V. V. Gribova, F. M. Moskalenko, V. A. Timchenko, E. A. Shalfeeva. Sozdanie zhiznesposobnyh intellektual'nyh sistem s upravljaemymi deklarativnymi komponentami. *Informacionnye i matematicheskie tehnologii v nauke i upravlenii*, 2018, no. 3 (in Russian).

[2] V. V. Gribova, E. A. Shalfeeva. Sistemy na osnove ontologicheskih baz znanij kak osnova dlja sozdanija sovremennyh sistem iskusstvennogo intellekta. *Vosemnadcataja Nacional'naja konferencija po iskusstvennomu intellektu s mezhdunarodnym uchastiem KII-2020*, 2020, pp. 12-19 (in Russian).

[3] M. Alizadeh, M. H. Shahrezaei, F. Tahernezhad-Javazm. Ontology Based Information Integration: a Survey. *arXiv preprint arXiv:1909.12372*, 2019.

[4] L. N. Lyadova, V. S. Zayakin, M. A. Smirnov. Formirovanie sobytijnyh rjadov s ispol'zovaniem mnogoaspektnyh ontologij. *Tehnologii razrabotki informacionnyh sistem TRIS-2020*, 2020, pp. 297-303 (in Russian).

[5] L. N. Lyadova, N. M. Suvorov, V. A. Vasiljuk. Arhitektura DSM-platformy, osnovannoj na znanijah. *Tehnologii razrabotki informacionnyh sistem TRIS-2020*, 2020, pp. 304-311 (in Russian).

[6] Ju. F. Tel'nov, V. A. Kazakov, V. M. Trembach. Razrabotka sistemy, osnovannoj na znanijah, dlja proektirovanija innovacionnyh processov sozdanija produkcii setevyh predprijatij. *Biznes-informatika*, 2020, vol. 14, no. 3 (in Russian).

[7] S. V. Pavlov, O. A. Efremova. Ontologicheskaja model' integracii raznorodnyh po strukture i tematike prostranstvennyh baz dannyh v edinuju regional'nuju bazu dannyh. *Ontologija proektirovanija*, 2017, vol. 7, no. 3 (25), pp. 323-333 (in Russian).

[8] M. Asfand-E-Yar, R. Ali. Semantic Integration of Heterogeneous Databases of Same Domain Using Ontology. *IEEE Access*, 2020, vol. 8, pp. 77903-77919.

[9] G. Xiao, D. Hovland, D. Bilidas, M. Rezk, M. Giese, D. Calvanese. Efficient Ontology-Based Data Integration with Canonical IRIs. *European Semantic Web Conference, Springer, Cham*, 2018, pp. 697-713.

[10] S. I. Chuprina, I. S. Postanogov. Koncepcija obogashhenija unasledovannyh informacionnyh sistem servisom zaprosov na estestvennom jazyke. *Vestnik Permskogo universiteta. Matematika. Mehanika. Informatika*, 2015, vol. 2, pp. 78-86 (in Russian).

[11] V. S. Kumar, P. Cuddihy, K. S. Aggour. NodeGroup: A Knowledge-Driven Data Management Abstraction for Industrial Machine Learning. *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*, 2019, pp. 1-4.

[12] OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) [Online]. Available: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/ [Accessed: 07-Apr-2022].

[13] V. V. Gribova, A. S. Kleshchev, F. M. Moskalenko, V. A. Timchenko, E. A. Shalfeeva. Rasshirjaemyj instrumentarij dlja sozdanija zhiznesposobnyh sistem s bazami znanij. *Programmnaja inzhenerija*, 2018, vol. 9, no. 8, pp. 339-348 (in Russian).

[14] L. N. Lyadova, V. S. Zayakin, M. A. Smirnov. Arhitektura sistemy analiza dannyh, poluchaemyh iz Internet-istochnikov. *Informatizacija i svjaz'*, 2021, no. 8, pp. 48-52 (in Russian).