Вычислительные технологии, ????, том ??, № ??, с. 1–10. © ФИЦ ИВТ, ????
Computational Technologies, ????, vol. ??, no. ??, pp. 1–10. © FRC ICT, ????
ISSN 1560-7534
eISSN 2313-691X

DOI:10.25743/ICT.????.??.??.???

Technologies of Semantic WEB as an environment of application development and integration

Evgeny A. Cherkashin[1,2,*]

[1]Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, 134 Lermontov St, 664033, Irkutsk, Russian Federation

[2]Institute for Mathematics and Information Technologies, Irkutsk State University, 20 Gagarina Bulv, 664003, Irkutsk, Russian Federation

*Контактный автор: Evgeny A. Cherkashin, e-mail: eugeneai@icc.ru

Поступила ?? ??? ???? г., доработана ?? ??? ???? г., принята в печать ?? ??? ???? г.

Semantic Web technologies and their standards are becoming productive assets for software development and integration on various design and implementation levels. These include OSI level 6 data representation, database and file storage formats, *etc.* They are also smoothly combined with the development and data analysis tools, giving rise to an environment for integrating distributed applications on a semantic level.

This paper presents a view on the technologies as data representation for software development tools based on model analysis, and transformation, examples are provided.

Keywords: knowledge graph, semantic web, model transformation, MDA, object-oriented logical programming, web application, GIS

Citing: Evgeny A. Cherkashin. Technologies of Semantic WEB as an environment of application development and integration. Вычислительные технологии. ????; ??(??):1–10.
DOI:10.25743/ICT.????.??.??.???.

1. Introduction

Since 2001 [1], the Semantic Web (SW) technologies have become actively used in describing software domains, representing semantic relations between objects. First-order knowledge theories are constructed for some domains based on the SW description, but in the practical aspect of the SW utilization, the solving data representation problems and software integration aspects prevail. Some tools for automation of conceptual model design and model conversion were proposed [2]. The R&D at present are carried on in developing applications, improving and standardizing vocabularies (conceptual domain models).

Unification of routine tasks resulted in standard development techniques for storing and processing data aggregated around the term “knowledge graph” (KG). These include data representation formats, vocabulary standards, query languages, tools for accessing and managing data in a KG. Software development techniques are mediated with properties of the

new-created resources and their APIs (Application Programming Interfaces). Many internet resources provide standard ways of formalized domain data access using query endpoints.

In Russia, domain model designers generally prefer to develop local vocabularies instead of spending time for the routine work of internet surfing for existing conceptual models and their enrichment. It is due to underdevelopment of the metadata resources (metavocabularies) due to lack of a domain classification system as compared, *e.g.*, to Universal Decimal Classification (UDC) used in the bibliography. Vocabularies also can have common parts, describing the same or similar entities with different terms and relations.

Unitizing standardized vocabularies (ontologies) in our projects allows us to compare other research groups' points of view to a problem with a formalized context and figure out underestimated and not fully formularized aspects of a domain under investigation or automation. The vocabularies are similar to information sources as scientific papers but have a concrete formal presentation of the information objects, allowing one to get rid of particular uncertainties. Another advantage of standardization is constructing a common ground of research, development, data formalization, and interchange formats and semantics.

This paper presents an author's experience of SW and KG utilization in the development of applications in a context of performance improvement of R&D. Three examples were reviewed, showing good points of the technologies' application.

2. Semantic Web tools

Briefly, SW technologies from a developer point of view include the following assets:

- Global identification of the entities via URI, IRI, URL ({Universal, International} Resource {Identifier, Locator}) are essentially the same notions but reflecting different representation properties of an identified resource;
- Conceptual set-theoretic and category-theoretic data representation notation as relations between resources and literals represented as `<subject, predicate, object>`;
- Triples used in a description of a domain form a corresponding semantic graph, conditionally subdivided into terminological subgraph (T-Box) and instances (A-Box);
- Graph presentation formats and techniques are implemented in files, databases, markups of documents, a data flow between applicants;
- There are databases comprising the generic domain objects descriptions, which are mostly A-Boxes (databases), for example, `DBPedia.org`, (formalized `Wikipedia.org`).
- For graphs being databases, query services are implemented to allow users to obtain its data by portions, as well as implement modifications;
- API and libraries for processing files and documents with semantic data acquisition.
- Software for graph data processing to improve its structure or induct patterns;
- Validation and verification services allow one to explore consistency and completeness of the conceptual domain models and their implementations in software.

Some assets form knowledge graph (KG) technological aspect of the SW that gained a particular interest within R&D in the last decade.

Each entity described in SW has its unique global IRI, its virtual or real internet address. Document entities' resource IRIs are formed accounting the document structure, *e.g.*, a hierarchy of contexts. Using IRIs in applications solves the fundamental problem of cross-domain and cross-application object identification and forms a basis for a standardized data processing environment. Entities of various abstract levels can be designated an IRI, including vocabulary definitions (*namespaces*), data types, data containers (files, databases).

Triples are a combination of two or three IRIs (`<subjectIRI, predicateIRI, object>`). Subject and predicate are always IRIs, whereas objects are either IRI (a reference to a resource) or a literal. Literals are values of basic types (*e.g.*, number, string, date, time) usually accompanied by metadata (type designator or a language tag). Types are defined in a vocabulary `<http://www.w3.org/2001/XMLSchema#>`, having well-known namespace abbreviation `xs`[1]. A language tag defines string values attribute, affiliating them to a cultural context. It is recognized with KG tools and utilized at presentation layers of the software.

Namespaces are convenient instruments in the triple definition. For example `integer` literal `42^^<http://www.w3.org/2001/XMLSchema#integer>` is reduced to `42^^xs:integer`, where `xs` is namespace prefix of a vocabulary having term `integer`. Most KG representation formats allow one to define a default namespace used without prefix. Almost any conceptual model of a domain represented as KG comprises entities taken from multiple vocabularies, and namespaces specify the origin of a term. Three-component tuple (triple) outside its KG context sometimes added the fourth KG IRI component (quadruple).

Much R&D efforts since 2001 [1] were spent to develop SW data formats and their coupling with document representation. The simplest way to provide a printed document with semantic representation is by pointing the user's software agent to a URL serving the semantic data. However, this approach requires implementing two different procedures for document generation and its metadata serving. That is why an RDFa[2] has been developed. It allows developer of XHTML[3] documents to enrich the document with its semantic layer. The semantic markup is represented with special attributes and their values. The hierarchical structure of XHTML is a matrix layer for the hierarchical structure of document entities' semantic relations. XHTML/RDFa processing libraries extract the layer. There are other formats having the same expression capabilities as RDF but used for other purposes: `N-triples`, `Turtle`, `Notation 3`, `JSON-LD`. The last is an efficient way of coupling SW and various generic data storage, indexing, and processing services.

Among published global vocabularies, `DBPedia` has the most value for our projects. It is used for concrete object or notion of physical world reference, *e.g.* the term "passport" in documents authoring [3], naming (labeling) referenced objects in a user interface with corresponding words in natural language. Another useful resource is `WordNet` converted in RDF triples. It describes the semantics of English words and various their mutual relations.

Program agents' access to the global resources is standardized as well. Serious vocabulary providers have an API endpoint of form `http[s]://<address>/sparql`[4] supporting direct `GET`-queries returning a user form, and `SPARQL POST` query with answering `JSON` encoded tabular result. The form is used to interactively test `SPARQL` queries before their incorporation in software agents. All `SPARQL` endpoints support federated queries, *i.e.* run subqueries on other servers. For a personal user KG storage, a number of servers were developed, *e.g.*, `GraphDB`, `Jena Fuseki`, `ClioPatria`. Each has its extension engines. `ClioPatria` is extended with Prolog modules and allows one to implement KG processing on server.

Formalized representation of a domain provided by SW techniques is the basis of automatic inference construction, thus modeling, *e.g.*, problem-solving, resulting in decision synthesis, theorem proving, domain verification. The inference is implemented in various ways. All of them belong either to semantic ones, usually implemented as solving a corre-

[1] Hereafter, all well-known namespace names will be written with a bold monospace font.
[2] Abbreviation of Resource Description Framework attribute language
[3] HTML structures with strict XML restrictions obeyed
[4] For example, see DBpedia access point at the URL `https://dbpedia.org/sparql`

sponding CSP (Constraint Satisfaction Problem), or syntax-driven procedures that mediate the previous structures and infer one of the specific properties (Automatic Theorem Proving), modifying a KG. Theoretic research is being carried on classification of vocabularies by the complexity of verification and development software implementing it for these classes.

Our interest is synthesizing (logically inferring) new objects using logic programming (LP) from domain models represented as KGs. LP allows a developer to combine KG processing with other inferences in the same programming paradigm. SWI-Prolog programming language environment and Logtalk extension is used for this purpose. SWI system has a library for representing and processing KG, and Logtalk is used to implement transformations providing the inference. Object-oriented properties of Logtalk enable one to cope with generic disadvantages of RDF with encapsulation, manipulate knowledge sets with object composition instruments such as extension, inheritance, and composition, and represent inference in a generalized form of objects sending messages to each other.

In order to present the general architecture of application development, consider briefly KG definitions and its useful properties. KGs changed the viewpoint to the knowledge base as being a part of the whole totality of knowledge, implying the obeying the techniques of its acquisition and processing [2]. The following outlines their definitions [2].

- Notions *data* and *knowledge* are converged into "*something known*";
- KG contains data, knowledge and metadata in the form of resources and relations;
- The same KG is being filled in and processed by different independent modules, *e.g.*, withing SPARQL queries with UPDATE statements;
- KG fundamentals allow developer to *postpone* the definition of a data schema;
- There are three points of view to graph schemata interpretation:
  - *semantic* aimed at domain modeling describing generalization relations (classes);
  - *validating* reflecting semantic verification, checking completeness with respect to sets of predefined general relations, and
  - *emergent* observed or deduced from exploitation experience analysis as a set of new generalized structures and *refactoring* of the KG.

In our projects, KG stores global data, knowledge, and models. It is due to general considerations not to overfill it with secondary plane data, which of cause can form other KG resources and be used in the federated queries. Such a way partially solves the main technical problems of KG storage and query execution. Being a triple set, whole KGs are usually stored in computer RAM, and their structures expose no heuristic information for query engines, which could be used for optimizations.

3. General architecture of application environment

With general constraints implied by KG based domain modeling, the design of application modules are to be built on standardized data interchange formats, which are to be easily converted from and to KG representation, as well as applications are to be constructed out of service modules supporting SPARQL endpoints whenever possible.

A convenient approach of application construction is usage component architectures, where subsystems in runtime are components *providing* interfaces using other components' service available via corresponding interfaces. Zope component architecture[5], Logtalk, React.js are used to *implement* interfaces. Python allows wrapping whole subsystems, *e.g.*,

[5]On-line book is at `https://muthukadan.net/docs/zca.html`

Elastic search, in imperative paradigm, and Logtalk is chosen for the declarative counterpart. Inter-process component communications is realized usually with `HTTP` protocol.

Our R&D resulted in a set of components and a general application architecture showed in Figure 1. The kernel module of applications is KG server (B2) built on the base of ClioPatria [4] with web-server and Pengines plug-ins. ClioPatria is a web-application implemented in SWI-Prolog being ISO-compliant Prolog implementation. KG functions are implemented with SW package, supplied in standard SWI-Prolog distribution. Pengines service enables SW web-applications to utilize logical inference on the server-side [5] with browser JavaScript environment integration. Pengine's knowledge base is extended with the programmer's Prolog modules. Kernel module stores a common global part of A- and T-boxes of applications, *i.e.*, it is application database. Other modules provide an application (C3).

Authoring tool (B1) is used to provide document generation, and integration [3]. This tool enables document content and structure inference in the browser utilizing templates and data from other web pages and websites providing LOD[6] [6] capabilities. Programmer implements a view of an object as a web page with an extended version of RDFa. The page is generated within any programming runtime, *e.g.* PHP. The authoring tool engine collects web page chunks and composes the document as an XHTML page. This composed page can be stored in a KG as a set of triples and indexed in the text indexing module (C1), printed with the browser's printing engine, or converted to PDF. The tool allows the user to change document content in WYSIWYG mode, its general structure is controlled by templates.

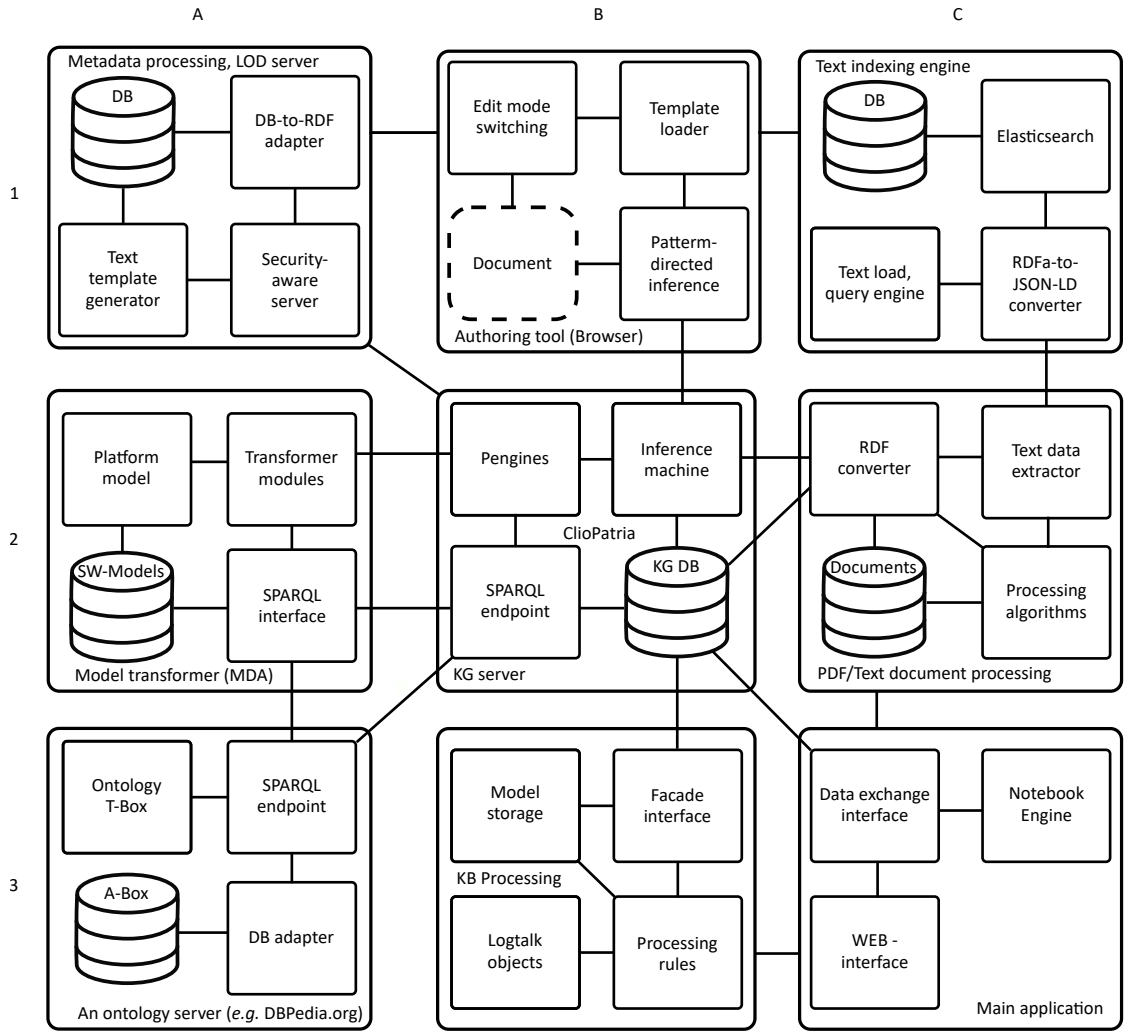
Text indexing engine (C1) provides a service for storing documents represented as KG triples, JSONs, and BLOBS of any format with corresponding text layers. This module has two implementations. One is built on the Sphinx Search engine, and the second is Elasticsearch. Elasticsearch supports JSON as the only document format and is easily integrated with KG using JSON-LD for triple representation. Sphinx Search is much faster in text and database records indexing and consumes less RAM as it is implemented in C++. The choice of an engine depends on the primary document representation.

BLOB stored documents having no markup (PDFs, scanned documents) usually contain valuable data to be revealed for application use. The documents could be report data or scientific paper content, or raster images. Recognition is implemented in the PDF/text document processing module (C2), where workers query the text indexing module for BLOB documents and add recognized text and high-order structure layers. All obtained layers are stored in the database of C1, and generally useful data are converted in triples of KG.

A generalized KG processing is being executed in the KB processing module (B3). This component denotes a set of rules used to implement the validating and emergent semantics and analysis and synthesis of the new data, including resultant output. Module A2 is the source of model data (design of domains, UML, problem statement, *etc.*), which is converted and form the T-box of KG. The conversion is a kind of Model-driven architecture (MDA) implementation. Transformation is carried out under the control of the Platform model, describing a context of the conversion. For example, in the case of software source code synthesis, the software platform model is used to implement model objects.

The metadata processing module (A1) denotes the environment capabilities for expressing the semantics out of the data processing modules' output and the input data semantics. It allows us to store resultant data in KG for further usage in module (B3) for problem-solving and decision-making. Module A3 denote external services and KG with valuable resources.

[6] Abbreviation of Linked Open Data



Abbreviations

MDA is Model-Driven Architecture, CIM is Computationally Independent Model, PIM is Platform Independent Model, PSM is Platform Specific Model, T-Box is Terminological Box, KB is Knowledge Base, DB is Database.

Fig. 1. Architecture of application instrumental environment as a set of components

4. Implementation examples

In this section, three examples of application implementation are being presented. The first example is an MDA application in synthesizing a Next-generation sequencing (NGS) investigation environment. After the invention of methods of NGS and their introduction in the practice of biological systems research, a new direction of molecular genetics is formed, which is referred to as *metagenomics*. The method allows biologists to describe many new groups of organisms at all taxonomic levels.

4.1. Next-generation sequencing

The aim of this R&D [3, 7] is to develop software support of the processes of NGS analysis with techniques and software for a visual representation of the computational process for so-called amplicon analysis so that domain specialists would be able to compose computational pipelines, which are executed on distributed heterogeneous computing resources (clouds). Later, the visual models will be used to organize the computational process, data store, and particular NGS data retrieval services.

The tools are based on dataflow representation of various stages of NGS standard operation procedures (SOPs) formed out of individual operations of Mothur library in Rapidminer studio Java application. To reflect Mothur's operations, we implemented source code analysis and transformation software. The C++ sources were analyzed by a Python syntax analyzer based on regular expressions and a top-level algorithm, generating a `TTL` source of KG representing Mothur modules' interfaces. That was a Computation-independent model (CIM) in the implemented MDA. CIM transformed into the Platform-independent model (PIM), represented Mothur's modules as dataflow modules, mainly differencing parameters to input/output dataflow ports and parameters. The Platform-specific model (PSM), *i.e.*, models of Java sources of the dataflow blocks, was constructed, in turn, from PSM.

All transformations were carried out with an object-oriented (OO) Logtalk programming environment representing modules that logically infer new data descriptions of the target objects. Thanks to OO, the transformation has two levels of abstraction in its representation. The higher one is a network of objects sending messages via their interfaces. At the lower one is analyzing and synthesizing rules, generating new conclusions on object properties.

The PSM objects must have an order in their description as they represent a model of source code to be generated. The order is organized with blocks of `assert`ed statement and subobjects sequences referencing KG elements. Blocks can be appended and pretended with new elements. At the rendering sources stage, the three blocks are traversed in-depth. This approach leverages the known `llvmlite` source code generation. This example of the application of KG utilizes strongly the property of data representation flexibility for description of the software object model data as well as one central point of data store and access. The disadvantageous properties of RDF, namely, the weak expressiveness with triples, was neglected by Prolog predicate and Logtalk object encapsulation.

4.2. Education document authoring

This project [8] is aimed at the construction of an environment for document authoring out of user-editable template elements, database query results, LOD-compliant web pages, and other document content. The target document is a web page where content is inferred while loaded by the browser.

All document elements are either *context* structures or templates filled with other elements. A context is an RDF hierarchical representation of a document data structure, providing data to be filled in a current rendered template. The document is rendered from its main template. All elements are SW resources. Templates can contain JavaScript subroutines for implementing special services, for example, table of contents out of CSS `class` labeled HTML headers of the generated document.

All elements are stored in a KG as triples and as LOD-compliant web pages, and according to SW considerations, all of them form a distributed KG. Data access differentiation by security model is implemented programmatically as the web pages disallow access to the restricted information. SW has special predicates for describing locations of a private data sub-KG in the open part. A namespace **taa** was added to define a markup interpreted as XHTML tree manipulation in the image and likeliness of open-source ZPT[7].

For document markup (document data representation), the following vocabularies are used:

- Friend-of-a-friend (**foaf**), agent information: individuals, legal entities, program agents;
- Provenance (**prov**), references between documents, data origins, proofs, *etc.*;
- Dublin Core (**dc**), web page annotation mark up;
- DBPedia resource (**dbr**), references to instant objects and classes;
- Schema.org (**schema**), Google, Yandex, Yahoo, *i.e.*, searchable objects, structural elements;
- The Bibliographic Ontology (**bibo**), literature reference mark up.

The general logical structure (sections, subsections) is represented with Open Annotation (**oa**) vocabulary originally used for annotation of already existing content. The generated document conversion into PDF and other formats is carried out under the control of **oa**. HTML5 default tag structures are used for visual sectioning and style definition. The **oa** vocabulary is also used to define parts of a document, which allow interactive editing after the document has been rendered. The obtained changes are fixed in a commit as new resources, and the containing document obtains new IRI fixing all the set of resource changes.

4.3. Web GIS on the base of a knowledge graph

Project [9] devoted to investigating capabilities of SW and KG technologies in representation spatially distributed data (GIS[8]-data). The fault data [9] are chosen as a subject for representation and publishing in this investigation. In geology, researchers accumulate data obtained after event observations, *e.g.*, earthquakes, landslides, by analyzing remote sensing data and results of field works. The obtained data are processed and interpreted, setting new attributes to a fault or refinement of their values. According to the geological research techniques, additional information is associated with attributes, clarifying their values. Such clarification comprises precision characteristics, measurement conditions, reliability assessment, paper references, and published fault data.

GIS represents spatial data in semantic-defined layers. For each object of a GIS layer, one can associate a set of attribute values of auxiliary data. The attribute set is the same for each object of the layer, regardless the attribute's value is defined or not. Empty values are represented as "`null`"s. In the case of geological exploration, when many attributes are undefined, this approach leads to sparsely filled tables. In this case, structure modification

[7] Abbreviation of Zope Page Template

[8] Abbreviation of Geographical Information System

and data analysis algorithms are required to utilize additional data checking stages when using standard relation operations (SELECT, UPDATE, DELETE).

Web publication applications, like information systems, implement filtering functions, differentiating the value attributes and their metadata. Screen widgets label names either defined in application configuration or figured out from the attribute names. Lack of vocabulary formal domain definition forces developers to spend more effort on user interface implementation.

In this project, we constructed a Web-application using contemporary Web and SW technologies for rendering a fault map, in which visual content is controlled by data filters implemented as SPARQL queries to KG representing fault data. For this purpose, data conversion from the tabular form to KG was carried out, a CliopPatria server for data storage was installed, and data viewers for the triple sets for a typical subject (a fault) have been implemented.

The proposed technology and software allow one to construct Web-GIS systems for research communities, as they support constant data accumulation, aggregation, and analysis thanks to the properties of knowledge graph (KG) data storage and processing.

Conclusion

Standardized forms of basic concepts of data representation and processing in Semantic Web (SW) research were considered in this paper. Special attention has been paid to reviewing Knowledge Graph (KG) data storage and processing in the scientific data processing software.

We did not consider analogous R&D as it was already reviewed in the corresponding publications. Another point of our R&D is improving existing technologies, data, and software with our tools, such as MDA. We present a review of obtained SW and KG technologies application experience. The main one is the powerful application integration capabilities, where all the set of implemented software forms a distributed environment of data processing. The main requirement is that the data representation at storage services and in the protocol implementations should obey SW and KG standards.

The main advantages of these technologies for the software developer are semantic markup and generic properties of KG, especially a possibility to postpone formal data structure definition as such declarative set-based data can be processed with a rule-based inference engine. Scientific community results in domain description in standardized vocabularies (ontologies), user communities aggregating and formalizing data on real-world objects form a productive background for problem space description and, in its turn, a common basis for application integration, which results in a cumulative effect of aggregation of developed software in a complex distributed data processing system and environments.

Further development of the technologies includes the following directions of great interest:

- Fibered representation of KG to support consistency and completeness check by reflecting A-Box entities and their relations to template domain models;
- Investigate expressive capabilities of Logtalk programming language as an instrument of knowledge (rules) representation for KG processing modules;
- Adding extended document analysis services [10] to the environment (section 3), which are developed in our research group;
- Improve integration techniques with metamodel presentation of the execution environment.

Acknowledgements. The results were obtained within the state assignment of the Ministry of Education and Science of Russia, the project "Methods and technologies of a cloud-based service-oriented digital platform for collecting, storing and processing large volumes of multi-format interdisciplinary data and knowledge based on the use of artificial intelligence, a model-driven approach, and machine learning", No. FWEW-2021-0005 (State registration No. 121030500071-2).

The results were obtained with the use of the network infrastructure of the Telecommunication center of collective use "Integrated information-computational network of Irkutsk scientific-educational complex" (`http://net.icc.ru`).

References

[1] Berners-Lee T., Hendler J., Lassila O. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American. 2001; 284: 1–5

[2] Hogan A., Blomqvist E., Cochez M., D'Amato C., *et al.* Knowledge Graphs. 2020. Available at: `https://arxiv.org/abs/2003.02320v5` (Accessed December 12, 2021)

[3] Cherkashin E., Shigarov A., Paramonov V. Representation of MDA transformation with logical objects. Proc. of "International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)". Novosibirsk, Russia; 2019: 0913–0918 DOI:10.1109/SIBIRCON48586.2019.8958008

[4] Wielemaker J., Beek W., Hildebrand M., Ossenbruggen J. ClioPatria: A SWI-Prolog infrastructure for the Semantic Web. Semantic Web. 2016; 7(5): 529–541 DOI:10.3233/SW-150191

[5] Lager T., Wielemaker J. Pengines: Web Logic Programming Made Easy. Theory and Practice of Logic Programming. 2014; 14(4-5): 539–552. DOI:10.1017/S1471068414000192

[6] Bizer C., Heath T., Berners-Lee T., Linked Data – The Story So Far. Int. J. Semantic Web Inf. Syst. 2009; 5: 1–22. DOI:10.4018/jswis.2009081901

[7] Cherkashin E., Shigarov A. Construction techniques of Baikal microbiome research information-computational environment. Proc. of "1st International Workshop on Advanced Information and Computation Technologies and Systems". December 7–11. Irkutsk, Russia. CEUR-WS.org; 2020. Available at: `http://ceur-ws.org/Vol-2858/paper2.pdf` (Accessed December 12, 2021)

[8] Cherkashin E., Shigarov A., Paramonov V., Mikhailov A., Digital archives supporting document content inference. Proc. of "42-nd International Convention on Information and Communication Technology Electronics and Microelectronics (MIPRO)". May 20–24. Opatia, Chroatia; 2019; 1037–1042. DOI:10.23919/MIPRO.2019.8757196

[9] Cherkashin E., Lunina J., Demyanov L., Tsygankov A. Web–GIS viewer for active faults data represented as a knowledge graph. Proc. of "4th Scientific-practical Workshop Information Technologies: Algorithms, Models, Systems". September 14. Irkutsk, Russia, CEUR-WS.org; 2021. Available at: `http://ceur-ws.org/Vol-2984/paper8.pdf` (Accessed December 12, 2021)

[10] Dorodnykh N., Yurin A., Shigarov A. Conceptual Model Engineering for Industrial Safety Inspection Based on Spreadsheet Data Analysis. In: Simian D., Stoica L. (eds) Modelling and Development of Intelligent Systems. MDIS 2019. Communications in Computer and Information Science. Springer, Cham. 2020; 1126. DOI:10.1007/978-3-030-39237-6_4