*УДК 004.652.8+004.91*

*Cherkashin E. A.,  
 т. PhD, senior researcher,  
Matrosov Institute for System Dynamics and Control Theory of Siberian Branch, Russian Academy of Sciences   
 tel. +7(914)870-67-54, e-mail: eugeneai@irnok.net,  
Orlova I. V.,  
 ф.-м. н., доцент,  
National Research Irkutsk State Technical University  
tel. +7(914)921-07-47 e-mail: soobshenie\_1@mail.ru*

**AN APPROACH TO CREATION OF LINKED-DATA BASED DOCUMENT ARCHIVES**

*E.A. Cherkashin, V.V. Paramonov, A.O. Shigarov, A.A. Mikhailov*

INSTRUMENTAL TOOLS FOR CONSTRUCTION OF DIGITAL ARCHIVES OF DOCUMENTS BASED ON LINKED DATA

*E.A.Cherkashin, I.V.Orlova*

**Abstract.** *We consider the problem of development of software tools for construction of digital archives of text documents. The main feature of the tools is the usage of Linked Open Data (Semantic Web) for representation and storing the logical structure of a document and its metainformation, logical inference system integrated in the server, and the tools of document composition from other parts. Linked Open Data allows us to extend the storage, indexing and retrieval functions of the documents with functions of data integrations with other web documents. An example of the instrumentation tools application for study course documentation authoring is considered.*

**Keywords:** *Linked Open Data, component architecture, digital archive, document authoring.*

Introduction

Technologies of Linked Open Data (LOD) [1] have been suggested by the W3C consortium to represent the semantic information in the data published in order to provide the possibility of its processing with software agents (Semantic Web), as well as to link all available information into a single semantic graph using relations and universal resource identifiers (URIs). The descriptive capabilities of the Semantic web technologies, services of HTML5 document serving and the LOD technologies create infrastructural basis of document publishing with providing logical links between documents. The document content information is associated with the content information of another one by means of relations (links) to relevant resources. Resources represent both the static contents and the results of the procedures execution publishing (converting) structural data, and generating marked-up text content.

An important advantage of using LOD in organizations' information environments is weakening requirements for storage of the information being published: the document itself is a data warehouse in a formalized way. To some extent, this fact reallocates the time spent on designing the database structure for storage partially structured documents to the process of solving a substantive problems: the user (developer) markups the text data with semantic structures with an instrument.

The aim of the study is creating tools for developing digital archives to design and implement a domain-specific repositories of information objects.

Based on the produced tool set, digital archives of documents are developed, which intended for generation of the documentation on study courses. The use of LOD data formats, web browser HTML processing, and developed technologies allows us to solve a wide class of problems of synthesis of documents, starting from the generation of substantial parts of the texts ending the presentation of stylistic characteristics of texts and the integration of logical markup into a global data retrieval services.

This work continues the research outlined in [2] in the direction of the software tools development including services storage, processing and access to semantic information.

Technologies of LOD

The technologies of LOD are based on the representation of the published information in the form of semantically marked-up document, usually web page or a result of a database query as HTML content. When using HTML for presentation of content, RDFa (Resource Description Framework in Attributes) is the primary mark-up format being supported. The markup is a graph (subgraph) formed by the relations (triples) of the form <subject, relation, object>, applied to a tree structure of HTML with special attributes. Most of these attributes are ignored by the web browser and do not affect the structure and style of the displayed document.

Separately from the visual content, standard Semantic Web formats RDF, OWL, TTL, N3 and 4, JSON-LD, RDF/JSON are used for presentation to the LOD. These representations are extensions of standard text, XML and JSON formats. There are several binary formats that support compact storage of the graphs in the file system.

The representation of a graph in workstation memory devices in most libraries and software frameworks are focused on fast search and filtering operations of triples according to one or more criteria, as well as the support of standard SPARQL queries. Some storage additionally support the interpretation of relations such as rdf:subClassOf, rdf:subPropertOf and relations derived from the rdfs:label. This allows one to create visualization of the contents of the graph, for example, to build the administrator interfaces for the repositories.

LOD developers created a number of resources to discover vocabularies (ontologies), including a set of base-level vocabularies for describing various aspects of the content (scos, rdfs, XMLSchema, dc, prov, oa, schema.org, etc.), graphs representing encyclopedic data, for example, Dbedia [3] and Wikidata [4], the Google Knowledge Graph, services of partial automation of semantic markup of text, such as Dbpedia Spotlight [5], means of the distributed execution of SPARQL queries. Created resources allow us to develop integration engines for data and graph distributed in Internet into a single information infrastructure.

Digital archive tools

Digital archive instrumental tools supporting LOD is being designed to provide development of publishing tools of data aggregated in an institution in a new form allowing its integration into the information infrastructure, e.g., in form of web applications publishing data of processing result in a scientific research.

The digital archive tool architecture is based on components. Zope Component Architecture (ZCA) [6] used in the Python programming environment [7] has been chosen as a architecture basis for the implementation of the instrumental tools. Rdflib library is used for the internal representation the LOD graphs. Graphs are serialized into appropriate text or binary formats when transferring data to an external process.

The toolkit consists of the following components:

* content and metadata repository with SPARQL and full-text search; the engines for LOD processing, based on the logical inference;
* service for analysis of the stored content enriching of the archive with semantic information describing the content;
* the tools for development of LOD applications and their user interfaces.

Storing LOD documents

The document content is stored either in the Kyotocabinet database, either as a file in a file system. Meta-information obtained by processing and analyzing a document is a content annotation and is described according to LOD standard vocabulary oa as a subgraph.

Each annotation (subject) is a tree with at least two nodes formed by the relations oa:hasTarget referring the original content and the oa:hasBody referring the annotation. The text content of the document, if it has been acquired, is the document annotation as well as a structure of type Dublin Core (dc) describing meta-information found. The contents (bytes, text) stored in Kyotocabinet is identified by its 128-bit Murmur3 hash function that allows us to create a reference key (hash) to the contents and backward. The hash is a content identifier in LOD. Used hashing system allows us to simply and effectively implement computational mapping at a sufficiently low probability of clash of hash values.

Let us list the basic properties of used vocabularies in description of the content.

Open Annotation (oa). The vocabulary standard is adopted in 2017, it is used to represent the content annotation describing other content. A browser bookmark is an example of such content.

Friend-of-a-friend (foaf) allows one to describe information about agents: physical and legal entities, software agents.

Provenence (prov). The prov ontology is the base of description of information flows in documents and their relationships. The prov vocabulary is used to associate the document and owner organization or physical entity.

In annotation, Dublin Core (dc) represents metainformation about the creative work: authors, format, contents, description, etc.

DBPedia resource (dbr) is the namespace of the objects (resources) of Wikipedia. The dbr ontology is used to refer to a specific entity, geographical names, etc.

Schema.org (schema) represents the objects that are recognized by the crawlers of Google, Yandex, Yahoo etc.

In addition to these ontologies, we use standard ontologies (RDF, RDFs, RDFa, XSD) and ontologies from the NEPOMUK project (https://userbase.kde.org/Nepomuk) to describe objects stored in the full-text indexes of digital archives.

Component of content analysis

Meta-information about the content is enriched with results of the analysis by so-called extractors. The extractors analyze the content and try to acquire information about text content, author, time of content creation, format, etc. Extractors are implemented with the utilities such as file, extract (libextract), tracker and recoll. All these utilities are free software and integrated with the component as external processes.

The results of the extractor analysis are presented as a subgraph oa, associated with the source content and stores in a metadata repository of LOD.

The metadata store

The implementation of metadata LOD graph storage is realized using ClioPatria [8] ontology server. The ClioPatria system is implemented in the SWI-Prolog programming environment. The project is under active development and aims to create a LOD graph storage service, implementing all the standards of storage and access, together with tight Prolog interpreter integration.

The system storage tier is implemented in C programming language. The store uses compact data structures to represent triples in RAM and on hard disk. A special feature of the storage is the fact that all triples are stored directly in RAM, as the service has no special assumptions about the structure of queries, which can be issued by adjacent software. Thus, server memory space is a critical resource of the repository.

ClioPatria developers implemented repository accessing in several ways: the SPARQL queries according to a standard protocol, direct queries to the ClioPatria Prolog subsystem using Pengines Protocol [9]. For the Pengines protocols a number of APIs has been created for object-oriented programming languages such as JavsScript, Java, Python. Using API classes, application developers and software agents are able to interact with the logic inference engine of the ClioPatria server.

The inference machine

Availability of ClioPatria's embedded programming system of SWI-Prolog [10] and the integration with RDFa allow one to process the marked-up digital information stored in the archive on the base of logical inference. Prolog allows us to make queries in SPARQL and Datalog formats, as well as to build a knowledge system to process the received data.

As experiment, in a very short time, we managed to integrate well-known Turbo Prolog 2.0 example of a natural-language interface to a database on the US geography with ClioPatria's built-in SWI-Prolog RDF support system. As a result, we developed a prototype with similar interface to a database of active faults of Siberia, aggregated at the Institute of earth crust SB RAS.

Another proof-of-concept application of the logical inference at the server site is the support of the development of a recommender system [11,12], based on the ontological description of the set of characteristics of the recommended object and the user characteristics. (FIXME: DID NOT IMPLEMENTED AT ALL and never be) The main problem to be solved in such systems is the generalization of stored data that allows to extend the set of derived recommendations with equivalent terms in the same general category, as well as the narrowing the retrieval result number by replacing general term with specialized ones more relevant to the search query.

Full-text search

The ClioPatria system has instruments of searching the text information for keywords, however, it is advisable to use a more powerful system of a full-text index, for example, Elasticsearch [13]. Implementation and integration of this service is quite simple, since every RDF graph is representable as JSON (JSON-LD), while JSON is the primary format for storing the indexed information in the ElasticSearch search engine; it is only necessary to separate a particular triples on the server, which are responsible for the representation of the resource to the user in the search results. Elasticsearch has the means of fuzzy comparison of terms that allows developing search subsystems retrieving relevant information. The main aspects of modules' function are discussed later in the article in the application section.

Application of the technologies

The designed tools used in solving a number of applied tasks built out of digital archive services, LOD and archive components.

The Ministry of education and science of the Russian Federation after a series of experiments switched to the large-scale incorporation of the Bologna process in the educational environment of the Russian Federation. One of the tasks solved in the framework of this introduction is the transition to competence-oriented representation of the requirements to the pedagogical process. The reform affects all aspects of the process, including the system of classification of specialties (introducing programs and directions, specialization according to skill levels (bachelor, master degrees, etc.), introduction of applied bachelor degree, list of courses, aims and objectives of the courses (negotiation with the competencies specified in the Federal standard), forms of conducting classes in universities (introduction of interactive forms of learning), the distribution of lectures and practical sessions, etc. The existing documentation is supplemented by new forms of mandatory documents: the Fund of Assessment Means (FAM), annotations of courses. In addition to this, the university management with the aim of improving the quality of educational services introduced its own additions to the form, content and requirements to the design documentation, in particular, to the quality of the conversion to HTML for publication on the website of the university.

For the minimum requirements fulfillment of the management of the universities for each course, the instructor is required to prepare at least three documents: the working program of the course, the summary annotation, and the FAM. This set is realized for each possible parameters combination of a course: the university, the department, specialty, direction (profile), programme, qualification (bachelor, specialist, undergraduate, graduate), academic or applied version of the qualification, form of study (full-time, part-time, evening, etc.). Each combination is reflected in the curriculum of the university from which data is annually or twice a year should be actualized in the work programmes.

The last five years showed the level of maturity of decisions made by the managers of the Ministry in terms of requirements to the presentation of the course - four generations of the standards (1, 2, 3 and 3+) are developed – for which it was required to resubmit documents in the appropriate updated form. The task of developing the new textbooks now seem more simple than before as compared to the fulfillment the course documentation requirements. Experience shows that most instructors are not able to cope with high-quality paperwork in the given time constraints, thus requiring the departments to hire a secretary, whose function is to bring documentation to the required quality level.

The solution to this problem lies in developing a software system that allows instructors to collect the texts of the work programmes, abstracts and FAMs from separate parts: a list of competencies and curriculum; contents (subjects of taught module/course) and FAM, which are shared between different versions of documents. The title pages are generated also from data of curriculum and templates set by the management. Microsoft Word and Excel means, that are commonly used for this task including built-in VBA, are clearly not enough to support problem solution.

Consider the scheme of construction of marked-up work program in the proposed system. The presentation format is based on the results in [14,15].

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
 <html lang="ru" xmlns=http://www.w3.org/1999/xhtml  
**xmlns:taa=http://irnok.net/engine/rdfa-manipulation** xml:lang="ru" metal:define-macro="page">  
 <head> <!-- Connecting stylesheets and modules -->  
 </head>  
 <body prefix="rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns# ... foaf: http://xmlns.com/foaf/0.1/ imei: imei.html#   
course: https://irnok.net/college/plan/01.03.02-16-1234-2461\_1%D0%BA\_PB-SM.plm.xml.xlsx-%D0%911.%D0%92.%D0%94%D0%92.3.1.html#"  
 resource="#post" typeof="schema:CreativeWork sioc:Post prov:Entity">  
<!-- The application control panel -->  
 <main lang="ru" resource="#annotation" typeof="oa:Annotation" id="main-document-container"><div property="oa:hasTarget" **resource="#course-work-program"></div>**  
 <article property="oa:hasBody"   
typeof="schema:Article foaf:Document curr:WorkingProgram"  
 resource="#course-work-program" id="main-document">  
 <div taa:content="imei:title-page"></div> <!--Титульный лист.. -->  
 <div taa:content="imei:neg-UMK"></div> <!--Approval sheet..-->  
 <section id="contents" class="break-after"> <h2 class="nocount c">Table of Contents</h2>  
 <div id="tableOfContents"></div>  
 </section>  
 <section id="course-description" resource="#description"  
**property="schema:hasPart" typeof="schema:CreativeWork">**  
 <div property="schema:hasPart" resource="#purpose"  
**typeof="dc:Text cnt:ContentAsText" >**  
 <div property="cnt:chars" datatype="xsd:string">  
 <h2 property="dc:title" datatype="xsd:string">Aims and objectives of the discipline (module)</h2>  
 <p>The aim of teaching the discipline "Technologies of programming" is the development of the following skills ...</p>  
 </div>  
 </div>  
. . . . . . . .  
 <div property="schema:hasPart" typeof="dc:Text  
cnt:ContentAsText" resource="#volume">  
 <div property="cnt:chars" datatype="xsd:string">  
 <h2 property="dc:title" datatype="xsd:string"> The volume of the discipline (module) and training activities (divided by type of training forms)</h2>  
 <div taa:content="course:time-distrib"></div>  
 </div>  
 </div>  
. . . . . . . .

In the example, the key structures are highlighted in bold font. Let us comment the structures.

The page displayed to the user is the abstract of the document "#annotation", and both the annotated content and the annotation text is the same at the stage of formation of the document – resource "#course-work-program". In LOD, all resources are global. This is achieved by substitution of the default namespace - the full URI of the current page - at the left side of the name of the resource.

The title page and the approval sheet are inserted from the templates page “imei.html” (information about the Institute of mathematics, economics and informatics of Irkutsk state university – math.imei.ru). The course data and the name of the specialty is filled in the templates from the context of the main document. All the key static templates of courses can be placed in one page of templates.

The text is divided into sections, surrounded by tags <div> and <span> with the relevant RDFa structures. Analysis of the experience of the developers LOD resources has shown that for the formation of the relationship is sufficient to use RDFa tags property, typeof and datatype. The use of rel and about attributes are recommended to be abandoned. This makes the structure of semantic markup more clean by reducing the number of entities used.

Team taa:content adds to the document a text from another HTML page, whose address is formed by the interpretation of the attribute, for example, taa:content="course:time-distrib" loads the text of a page generated by the table of distribution of time spent between the types of classes (lectures, practice, laboratory, SRS, etc.). To generate such tables, a web server has been realized for serving web pages of these courses. Each page represents a course/module, scheduled in the curriculum. The page content is encoded by the structure of its URL and rendered at the time of first request. Text templates are identifier by the id attribute.

In edit mode, the work program's tags combination with attributes property="cnt:chars" and datatype="xsd:string" denote conversion of their texts to editable form. For clarity, the document without taa:-inclusions is shown to the user, indicating these inclusions with the special tags and styles. At the end of editing, the text is stored in the server file system. The user can then commit the changes and sync the text of the saved document to the server control versions (git).

In another application, we implemented the service of automation of the generation of documents with the grammar agreement engine substituting the phrases and suffixes in its text. Consider the example of power of attorney form representations (in Russian).

<p>Я, <span property="fibol:designatesSignatory bibo:owner"  
 typeof="fibol:Signatory foaf:Person dbr:Principal"  
 resource="#principal"><span property="foaf:name" id="signatory-name" datatype="xsd:string" class="edit">Иванова Елена Викторовна</span>,<span property="adoc:hasPassport" resource="#signatory-passport" typeof="acrt:Certification"> <span property="acrt:qualification" resource="dbr:Passport">паспорт</span>…

This example expresses the fact that the principal (fibol:designatesSignatory) is certified (adoc:hasPassport) with a passport (acrt:qualification dbr:Passport).

In the edit mode of the document, tags with the attributes datatype="xsd:string" and class="edit" are converted into the input fields. The values of these fields propagate further in the text of the document. Harmonization of grammar is implemented with the an id, data- and class attributes. Attribute class=“disp” specifies the substitution of a phrase identified by id, in the text of the tag. The attributes data-m and data-f specify the variant of a word with respect to the genus of the corresponding noun. The attribute data-case specifies the declination (case and number) for words being filled in in the text. The grammar transformation algorithms are implemented on the server.

Let us continue the previous example: processing grammatical structure.

...проживающ<span class="disp" id="signatory-name" data-m="ий" data-f="ая">\_\_\_</span> по адресу:... Подпись <span class="disp string" id="signatory-name" data-case="gent">\_\_\_</span> удостоверяю...

The project is available as modules of the Python programming language at: https://github.com/isu-enterprise.

Conclusion

The paper discusses the facilities of the use of Linked Open Data (LOD) technologies for the solution of tasks of creating digital document archives with instrumental tools for creating new documents based on the stored data. Layout is carried out using algorithms that interpret the relationships between the elements of the HTML tree. The layout algorithms are triggered in the presence of corresponding conditions in the nodes of the document tree. Each algorithm makes changes in the tree, resulting the final content and styles of the document. Forming documents are stored on servers that support the HTTP Protocol.

An example of use of the development tools for the document synthesis in the educational environment and legal documents has been presented. Further development of this project is carried in the following directions: a) improvement of the tools of typesetting documents, b) automation of the layout based on analysis of changes in the documents [2], c) implementation in practical applications, d) collecting of additional information about user needs, d) develop a regulated access to information stored in digital archives. The created software tools and technologies aimed at the development of a software infrastructure of a global electronic document flow, allowing individuals and organizations to share documents, whose logical structure is partially formalizable within the framework of the traditional software of document automation.

Acknowledgments

The results are obtained with the active use of the network infrastructure of Telecommunications of Center of collective use "Integrated information-computing network of Irkutsk scientific-educational complex" (http://net.icc.ru).

REFERENCES

1. Bizer Ch., Heath T., Berners-Lee T. Linked Data – The Story So Far // International Journal on Semantic Web and Information Systems. 2009. Vol. 5 (3). P. 1–22.

doi:10.4018/jswis.2009081901.

1. Cherkashin, E. A., Belykh, P. V. et al. An approach to managing a site content on the base of RDF technologies // Materials of All-Russian conference with international participation "Knowledge – Ontology – Theory" (KONT-2013), 8 – 10 October. 2013. Vol. 2. Novosibirsk. Publishing house "AIC Price-courier".

ISSN 0568-661X

1. Lehmann J., Isele R., Jakob M., Jentzsch A., Kontokostas D., et al. DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia // Semantic Web Journal. 2015. Vol. 6, No. 2, P. 167-195, IOS Press.
2. Krötzsch M. How to use Wikidata: Things to make and do with 40 million statements // In Keynote at the 10th Wikimania Conference. 2014.
3. Daiber J., Jakob M., Mendes P. Improving Efficiency and Accuracy in Multilingual Entity Extraction // Proceedings of the 9th International Conference on Semantic Systems (I-Semantics), 2013.

URL: http://korrekt.org/talks/2014/wikimania-wikidata.svg (access-date: 01.07.2017)

1. Baiju M. A Comprehensive Guide to Zope Component Architecture. [Electronic resource]

URL: http://muthukadan.net/docs/zca.html (access date - 01.07.2017)

1. Langtangen H. A Primer on Scientific Programming with Python (Texts in Computational Science and Engineering) 3rd ed. Springer. 2012. 798 P.
2. Wielemaker J., Beek W., Hildebrand M., Ossenbruggen J. ClioPatria: A SWI-Prolog infrastructure for the Semantic Web //  Semantic Web. 2016. Vol. 7(5). P. 529-541, DOI: 10.3233/SW-150191
3. Lager T., Wielemaker J. Pengines: Web Logic Programming Made Easy // Theory and Practice of Logic Programming. 2014. Vol. 14(4-5),

DOI: 10.1017/S1471068414000192

1. Wielemaker J., Schreiber G., Wielinga B. Prolog-Based Infrastructure for RDF: Scalability and Performance // In: Fensel D., Sycara K., Mylopoulos J. (eds) The Semantic Web - ISWC 2003. ISWC 2003. Lecture Notes in Computer Science. 2003. Vol. 2870. Springer, Berlin, Heidelberg.
2. Nefedova,Yu. S. The architecture of hybrid Recommender system GEFEST (Generation–Expansion–Filtering–Sorting–Truncation) // Systems and software of Informatics. 2012, vol. 22, issue.2, pp. 176–196.
3. Beel J., Gripp B., Langer S., Breitinger C. Research-paper recommender systems: a literature survey // International Journal on Digital Libraries. 2016. Vol. 17. P. 305. doi:10.1007/s00799-015-0156-0. (access date - 12.12.2016)
4. Kuć R., Rogoziński M. Mastering Elasticsearch - Second Edition. Packt Publishing. 2015. 372 p. ISBN-9781783553792
5. Capadisli S., Guy A., Verborgh R., Lange C., Auer S., Berners-Lee T. Decentralised Authoring, Annotations and Notifications for a Read-Write Web with dokieli // Procs of ICWE international conference, 5-8 June, 2017, Rome, Italy. (to appear) [Electronic resource]: Preprint URL:http://csarven.ca/dokieli-rww (FIXME: PAGES)
6. Heino N, Tramp S, Auer S, et al. Managing Web Content using Linked Data Principles – Combining semantic structure with dynamic content syndication // Computer Software and Applications Conference (COMPSAC), IEEE 35th Annual. 2011. P. 245-250. [Electronic resource] URL:http://svn.aksw.org/papers/2011/COMPSAC\_lod2.eu/public.pdf (access date - 30.05.2013).