A Document Content Logical Layer Induction on the Base of Ontologies and Processing Changes

Evgeny A. Cherkashin*, Polina V. Belykh*, Danil V. Annenkov** and Christina K. Paskal**
* Institute of System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
** National Research Irkutsk State Technical University, Irkutsk, Russia
eugeneai@icc.ru, polinka_irk@mail.ru, annenkov@ib-soft.ru, paskal@istu.edu

Abstract – An approach to a document content representation on the base of ontologies is considered. The ontologies are written in popular RDF (Resource Description Framework) format and its variations. Technological aspects of document rendering in internet browsers are considered, as well as Chameleon template HTML generator improvement to support the rendering. Other aspects such as multiformat data representation, a modification-driven data and knowledge acquisition are presented. The technologies are applied in notary office automation for preparation new documents. Application of RDF and its processing algorithms allowed us to solve common problems of logical layer document representation and management of the document content.

I. INTRODUCTION

In 2001 Tim Berners-Lee proposed [1] a blueprint of further web development that is aimed at the development of network services with reasonable integration of logical layer of the information stored in the network. Information presented on sites is marked up semantically and program agents use the markup as the logical layer for the information consumption and processing. The blueprint is referred to as Semantic Web (SW).

One of the main problems of SW is the fact, that the regular users of the web resources are not fond of the technological aspects of Semantic Web. They are interested in their practical problems solutions. In order to involve users in SW content development we must completely hide the technological aspects from them. This results in the necessity to develop document content management software exploited within SW as knowledge acquisition systems, where user plays role of information source for decision-making engine.

The content of a legal document in most cases contains meaningful information for human that is usually passed to other documents in a derivative form. This suggests an idea to develop a form of the logical layer representation in a low level form, which can be rendered by means of context-depended (in sense of document) templates. The SW approach could supplement the idea with data formats and technologies. SW represents logical layer as a network graph of notions and relations between the notions. For example, individuals mentioned in a document relate to the document as “parts-whole”, unless to speak even more explicitly.

At present most of the ontology models of domains used for refining search results. Automatic procedures of ontology extraction from the documents are based on crawling documents in a warehouse and data mining procedures on the text bodies and metadata attributes of the documents. Simple observation on the human behavior in the process of a document preparation will result in the confidence that meaningful parts of the document are located at the points of document modifications. Hence, a programming system automating a document preparation can track the modifications and analyze them to extract ontology data, i.e., data and knowledge.

The logical layer is induced during content modification by means of data analysis and user interview. The context of the knowledge acquisition consists of the source document (its textual representation and the logical structure), its list of modifications in a transaction, user's action history and answers to the interview questions clarifying meaning of his/her actions. As a result new or modified triples <subject, relation, object> representing the domain are constructed. Collected data and metadata of the logical layer can also be crawled on a regular basis to figure out patterns and functional relations between triplet data. In the last case a relational table could be constructed to raise the efficiently of data storage and processing.

At present, there are two approaches for semantic text markup: a) joint use of HTML and RDF (Resource Description Framework) specifications (RDFa) for the semantic content definition; b) special interpretation of HTML attribute combinations without use of additional extensions of HTML.

The first approach is a result of theoretical development of the SW basis. A number of semantic representation language classes are formed according to the language expression abilities, complexity of the processing algorithms, and decidability of inference process in the corresponding description logics. The second approach is aimed at a guaranteed decidable algorithmic processing of the semantic markup. Technology of microformats [2] is the most widely known example of the approach. Microformats are processed with browser plug-in modules. The common element of the two approaches is the fact that HTML page, which is a

presentation layer of the document, is a carrier of the semantic (logic) layer of the data to be visualized.

The present trends of internet information system development shows that the systems become a web services and are oriented to support social networks [3]. The data flow processing here in most cases is input, storage, filtering and transmitting data (i.e., data integration) rather than aggregation and report generation. The usage of special design techniques for the middleware layer of the software, such as object-oriented design, is not a significant advantage. Social network are interconnected with common protocols and data structures, also in the environment

- there are no predominant common task to be solved with all the agents;
- each agent solves its special task, so the agent API's and supported data format must be strongly standardized;
- human users of the social network do the aggregation tasks personally including aggregation of unstructured information.

In the paper, we consider an approach to document content management and integration, including web-site content, based on RDF standards. The human user plays a role of a data source in the process of semantic data markup formation of the document content.

As a testing ground, we have chosen document preparation automation of a notarial office. Most of all operations over documents can be expressed as textual and logical layer modification, for example, data of the logical layer are copied from one document to another; sometimes roles of individuals mentioned in the documents are alternated; database collects client data for further reuse, etc.

A part of the paper devoted to consideration of organizational problems, such as involving knowledge engineers in a refinement process of generated parts of the ontologies; partial automatic ontology verification; implementing secure ways of personal data transfer and processing. The properties of the document exchange network will be similar to social networks, and, probably, can be further developed and investigated the same way.

II. DOCUMENT CONTENT REPRESENTATION

The RDF standard describes informational resources as triples <subject, relation, object> in a context. Each set of triples forms a graph (network) of data and relations reflecting knowledge. It is convenient to divide graphs to subgraphs and construct their hierarchic complexes [4], resulting a hierarchy of contexts. In a general case, a context affects to the interpretation of its set of triples. For example, family name and passport data are presented as texts in different parts of the document, but related to the single person in the context defined by the document.

In the discussed approach, all data for document rendering is also stored in the ontology graph. We represent with triples the views and algorithms implementing controllers in the sense of MVC (Model View Controller) technique of user interface design model [5]. Let us consider an example of document context (contact data of an individual) and its rendering environment using FOAF (Friend of a Friend) ontology format.

```
<rdf:RDF
 xmlns="http://www.w3.org/1999/xhtml"
 xmlns:rdf=
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:foaf="http://xmlns.com/foaf/0.1/"
 xmlns:s="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:view=
  "http://purl.org/aquarium/engine/MVC"
 xmlns:tal="http://xml.zope.org/namespaces/tal">
  <rdf:Description rdf:about=
  "http://www.example.com/People/II/contact#me">
    <rdf:type rdf:resource=
     "http://xmlns.com/foaf/0.1/Person"/>
    <s:seeAlso rdf:resource=
     "http://www.exam..../People/II/contact"/>
    <foaf:homepage rdf:resource=
     "http://www.example.com/People/II"/>
    <foaf:img rdf:resource=
     "http://www.example.com/people/II.png"/>
    <foaf:mbox rdf:resource=
     "mailto:ii@www.example.com"/>
    <foaf:name lang="en">Ivan Ivanov</foaf:name>
    <foaf:name lang="ru">Иван Иванов</foaf:name>
  </rdf:Description>
  <!– Render a Person into HTML+RDFa. -->
  <rdf:Description rdf:about=
   "http://xmlns.com/foaf/0.1/Person">
    <view:pt xml:lang="ru">
      <--! Variable "subj" contains subject. -->
      <a href="." tal:attributes=
       "href rdf: subj s:seeAlso"
       tal:omit-tag="not: rdf: subj s:seeAlso">
        <span tal:replace=
         "rdf: subj foaf:name"/>
      </a><br/>
      Home page: <a href=""
                  tal:attributes=
                 "href rdf: subj foaf:homepage">
       <span tal:replace=
        "rdf: subj foaf:homepage"/>
      </a><br/>
      E-mail: <a href="mailto:me@example.com"
                tal:attributes=
          "href string:mailto:
           ${rdf: subj foaf:mbox}">
      <span tal:replace=
         "rdf: subj foaf:mbox"/></a>
    </view:pt>
  </rdf:Description>
</rdf:RDF>
```

Note, that the first part of the example defines an instance (a contact data of Ivan Ivanov) in the ontology. The second part of the example describes a view for the resources that are instances of class Person. A modified version of Cameleon rendering engine is used as document template subsystem. Chameleon transparently interprets object values and triples as replacement of the source HTML tree branches and leaves. The set of global

variables passed to a template contains variable `subj`, which defines subject to be rendered, `container`, which refers to context graph, and template, which refers to the view itself. In the example `subj` contains an instance of Person class, context contains document, where the instance is mentioned. The `template` reference is used to store, constant auxiliary data that helps rendering.

In order to support rendering triples Chameleon engine was improved also with new syntactic rules corresponding to description logic expressions influenced by RDF standard (standard identifiers, XML namespaces, *etc.*). The renewed engine marks up resulting HTML with the logical layer data by means of RDFa-structures. The improvement allows us to transfer the document and its logical layer in the single data stream, as well as supply the browser-side routines with necessary logical data. The transferred logical layer, then, is interpreted by context-dependent editing widgets. User points mouse to a subject of a triple in the displayed document and a corresponding widget appears. For example, pointing to family name of a person a string field appears to change the value. The result of a modification is immediately sent to the server.

In a general case, it is necessary to implement functional relations algorithmically for view. This is realized with programming language inclusions in the RDF definition of the view resource. In the following example, a function implementation for class Person is presented.

```
<rdf:RDF xmlns:html=
 ”http://www.w3.org/1999/xhtml”
  xmlns:rdf=http://www.w3.../22-rdf-syntax-ns#
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:s=http://www.w3..../rdf-schema#
  xmlns:view=
   ”http://purl.org/aquarium/engine/MVC”
  xmlns:tal=
   "http://xml.zope.org/namespaces/tal">
  <rdf:Description rdf:about=
   "http://xmlns.com/foaf/0.1/Person">
    <view:method html:id=”nameUpcased”>
       <html:script
        type="text/python" xml:lang=”ru”>
        <--! The instance defined as “self” -->
         return self[“foaf:name”][0].upper()
       <html:script>
       <html:script
         type="text/javascript" xml:lang=”ru”>
        <--! The instance defined as “this” -->
         return this.query(“foaf:name”)[0] \
            .toUpperCase()
       <html:script>
    </view:method>
  </rdf:Description>
</rdf:RDF>
```

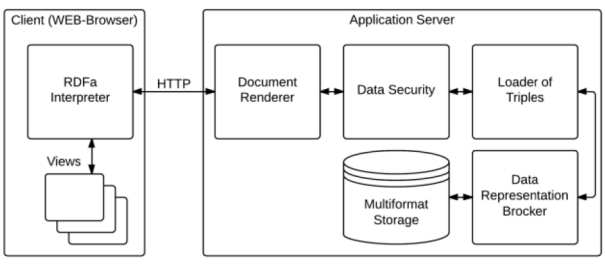

Figure 1. Program system architecture

Using these two definitions, template engines are able to take advantage of the method “nameUpcased” (implemented in various programming languages) to make a string argument as upper case. Role based access control can be implemented the similar way [6, 7] using RDF triples.

In the fig. 1 architecture of the document content management system is presented. The system reflects popular client-server architectures, where the server-side interprets Chameleon template (module “Document Renderer”). In order to render a template the data model instance should be loaded through module “Loader of Triples”. The instance is represented as a set of triples; the resulting document shows only data allowed to view to user. The data access is under control of “Data Security” module. The restricted triples are filtered out, and the fact is noted in a special document field. The purpose of “Data Representation Broker” module is to store triples in a format that is better support special tasks, such as aggregations. Three formats are supported – RDF, XML, and relational tables. In the backward direction, the broker restores triples from the storage as RDF entities. All these storage engines are implemented in OpenLink Virtuoso Universal Server [8] as well as its Open-Source Edition.

III. AN EXAMPLE OF APPLICATION

One of the applications of the technology under development is document preparation automation of a notary office. Notary office in Russia generate vast amount of printed documents. The documents contain both formalized and unformalized data. For example, formalized data are the passport data of individuals, registration data of vehicles. Data of this kind are passed from one document to other documents, raising a bindles of related documents. The document content structure significantly depends on entry fields’ data filled in: form of notary signature field depends on legal capacities of the individuals mentioned in the main document body. At present templates of the documents are prepared by a programming engineer with collaboration to a professional secretary. Unformalized data are various enumerations of legal empowerments of the individuals, article codes, and explanatory text.

Application of the semantic approach to the logical layer representation of the notary document content is aimed at involvement of the secretaries (people with low engineering skills) into the processes of the document instances preparation from the existing templates and into the development of the templates and constructing their hierarchical arrangement.

Logical layer of a notary document consists of hierarchy of subjects. The subjects relate to each other on various abstraction levels, e.g., in a letter of attorney there are at least two individuals, the first one is principal, and the second one is trusted (proxy). For each individual passport, data and place of residence are defined. The letter can contain other person data, *e.g.*, for partially capable children. All the individuals are in explicit relation to the document. The triplets `<letatt998`

`containsPrincipal indiv_78>` and `<letatt998 containsProxy indiv_79>` could define a principal and proxy to the document. The mentioned relations are specifications of abstract `hasIndividual` and `structuralElement` relations. This multilevel approach allows us visually represent hierarchical structures of the document, having interpreted the relations as structural elements, and in the same time to associate individuals with other roles in new documents, swap roles during editing. The swap function is implemented as user interface widget, which recognizes certain subject-to-subject relation patterns in the document body and interprets semantics of the relations as possible action. A general graph structure of the abstract level of a notary office automation ontology is drawn in fig 2.

Semantic markup can be used to define grammar transformation of natural language phrases, nouns and adjectives, including family names of individuals. For the purpose a structure `view:decl` (a shortcut of declination) was introduced, which each noun and adjective of the tag text transforms into required declination and form. For coordination of nouns and adjectives expressions of a form `<... view:gender="EXP M F N">TEXT</...>` are used. The interpreter appends to text `TEXT` string `M` if expression `EXP` is a masculine noun, string `F` if it is feminine, and `N` for neuter. Variables `EXP, TEXT, M, F, N` are string expressions, `N` is optional, `M, F, N` are divided by one space symbol or with "_".

IV. Content Management and Ontology Markup Construction

One of the key moments of the research is semantic document markup techniques adaptation to the process of document editing. We suppose that a document body modification, in a regular case, alters the meaning of the document, hence, alters its logical layer. Examples of the modifications are the elementary error correction, a field value change (object of a triple), paragraph text editing that might imply its origin template correction. Each modification also will result to a new relation (triple) extraction between context subjects and old/new object value. The text modification analysis is aimed at data and knowledge acquisition, where the content management system plays an active role, and user is a source of complementary information.

The semantic layer enrichment is carried on in an environment containing logic information thanks to document is already supplied with a logical layer. The source data for an acquisition step is as follows: a) the source version of the document, b) a text modification

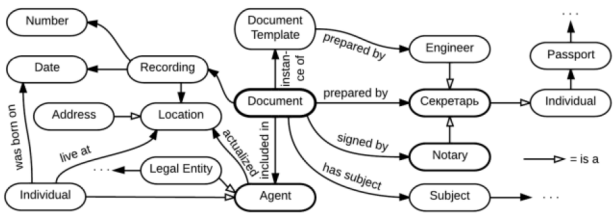

Figure 2. A general structure of notary document ontology

expressed in `diff` format [9]; c) user answers to the question asked by the system, which refine the semantics and structure of the modifications; d) user actions preceding the modification. If the modification corresponds to a value of an object of a triple then the modification means either error correction (subject is not changed) or creation of a new subject if user copied the document previously. Creation of a new document implies filling in a number of triples with new values representing a new subject (RDF resource) of the document.

If a field or a text is partially modified this can be interpreted as:

- again, an error correction without document meaning change;
- refinement of the logical structure of subject, extraction of a relation and an object; this corresponds to a new triple connecting the subject to the new object (changed value); the user must choose the subject of the new triple.

The new subject is chosen form a tree of all the subjects of the edited document. After the choice is made, a list of all available relation is constructed from all known relation of the subject, its class and parent classes. User must choose one relation to form the triple. In the case when the list contains no desired relation a new one must be defined. New relation is always a subclass of one that already exists in the system, which is also chosen from the list. New relations defined by inexperienced user must be periodically analyzed by knowledge engineers to get rid of semantic inaccuracy, contradictions, redundancy to the equivalents, and be, hence, refactored.

If a value of a triple is removed, then the triple is being removed too. The situation is acceptable if the minimal structural and semantic completeness of all the subjects of the documents are hold, otherwise either the user delete action is prohibited or the chain recursive deletion of the subjects is initiated. To control this behavior each subject class have to be accompanied with a list of minimal valuable triplets that define the basis of the subject sense. Partial text removal, if it is not an error correction, is processed analogously to modification, with removed part being the object value. Addition of characters to text is an action similar to modification, i.e., in a general case a new triple is constructed.

Addition of a triple might result in extending the document with new subjects and relations. For example, let construct a new tripartite agreement from an existing bilateral one. In new contract a third individual appeared, so the addition of a new family name of the individual results in construction of a subject for the individual, filling in the necessary triples, as well as definition of a new relation between document and the subject as a subclass of `structuralElement` relation.

A. *Ontology consistency check*

As in passing mentioned above the user will be a cause of ontology inconsistency. Class inheritance allowed in popular ontology formalisms may be the reason of contradictions, *i.e.*, on an abstract layer it can appear that

an individual belongs to both of disjoint classes. Multiple inheritance may result to semantically incorrect relation inheritance: one can device a dish as subclass of pizza and ice cream [10].

To track the anomalies of the ontological descriptions, which appear as empty classes (which have to be non-empty), and contradictions as in the first case, a verification subsystem is being implemented. The subsystem is a SWI-Prolog [11] program importing a context graph or a subject as RDF triples and running a number of tests over the triples. The result of the verification is a list of potential problematic contexts, subjects and triples.

The verification subsystem is constructed by students as their laboratory works. As the test case a pizza ontology developed at the University of Manchester [10] is used. The correctness of the subsystem implementation is being compared with Java semantic checker integrated in last version of Protégé system [12].

V. Related projects and further development

Among existing projects of ontology based document management the project "Semantic MediaWiki" can be emphasized. A basic Wiki text editing is extended with semantic annotations represented in a special markup format. The annotations are used by search engine of the Wiki pages [13].

In comparison to Semantic MediaWiki the project OntoWiki [14] obtained the similar results from an opposite starting point. OntoWiki is based on the predominantly usage of logical representation of the information as semantic network. The logical structure is edited with entry forms generated on the base of predefined vocabularies (term sets). User is allowed to change just one text property `lod:content`, which can contain HTML markup. The HTML markup does not relate to the logical structure of the subject to be visualized as web-document. The text is modified with WYSIWYG editor integrated into OntoWiki. The project is aimed at social networks developed under control of Linked Data technology.

Our project can be positioned as a further development if the OntoWiki engine to support a natural representation of the document content, visual editing of the content, conserving the logical layer; implementation of the data and knowledge acquisition on the base of modification analysis. The templates for OntoWiki subjects rendering are stored beyond the ontology and separately from the main content of the document. In our case the templates are auxiliary elements of ontology, it can be logically inferred from inheritance. Most of the interrelations between text content and its logical structure are expressed in RDFa.

A further development direction of the project is aimed at implementation of cognitive data mining (data analysis) in similar documents to reveal patterns between attributes that appear in the documents. The results of the analysis may be a basis of automation of document data storage technique decision. For example, if a set of attributes shows a strong correlation to an attribute than it can be interpreted as a relation of a relational database. The set forms a determinant (the key set of a table) and the set of strongly correlated attributes forms rest of the table as described in the Method of Functional Dependency. The method used by database engineers in the process of a database design, the dependencies revealed from the analysis of the attributes' semantics. That is, the method could be used conversely, the dependencies can be revealed on the base of data mining.

The abstract layer of information modeling of the knowledge acquisition is a category of system complexes (configurations) [15], which is perfectly embody common metamodel of the ontologies as well as supply additional structural and functional properties. Developing the theory further one can connect the ontology devised during the document preparation process to the stage of UML-modelling of information system, which automates the processes of the domain.

Another directions of the system development is a reduction of HTML-rendering engines of the logical layer, so it could be completely realized on client side, as well as implementation for other programming languages and software platforms. Also we plan to develop a plug-in module for OntoWiki allowing editing Wiki interactively.

VI. Conclusion

An approach to representation of logical layer of a document based on RDF (Resource Description Framework) is proposed. The approach allows us to formalize the structure and semantic relation of the document, and also store data to render the document as HTML-page in the same data format - RDF. XML and RDF allow us to join logical and presentation aspects of the document within the same storage engine. The engine stores data as an onology, *i.e.*, set of triples `<subject, relation, object>`. A technique for HTML-rendering from the logical layer is described. The resulting document will contain the logical layer as a RDFa markup. The generated RDFa-markup is used at client side by web-browser for control of WYSIWYG-editing of the document. Text elements are modified with special widgets appearing in the user interface on an mouse event. A technique for organization of an interactive process of logical layer forming of the document content on the base of modifications analysis of the document content introduced by user. An example of application of the technologies under development in a notary office is presented. Thus, we shown that RDF format mixed with XML allows us to represent logical layer of meaningful information of a document, as well as sharing common data between documents.

On the base of the technology a network of document data exchange can be devised. The security of the document transmission can be provided as off-line data streams: each physical document is accompanied with its bar- or QR-code encoding the corresponding RDF-data of the transferred document. This can result in a semantic network analogous to nowadays social networks.

Acknowledgment

The research is carried on under support of Integration multidisciplinary project of Siberian Branch of Russian Academy of Sciences N 17 “Development of services and infrastructure of scientific spatial data for supporting complex multidisciplinary scientific research of Baikal nature territory”

REFERENCES

[1] T. Berners-Lee, J. Hendler, O. Lissila. The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities, Scientific American, May 17, 2001, pp.1-18. URL: http://sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21. (access date: 05.09.2013).

[2] Microformats. URL:http://microformats.org/ (access date: 30.05.2013).

[3] Social network - Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Social_network (access date: 20.08.2013).

[4] Chameleon – Chameleon 2.10 documentation. http://chameleon.readthedocs.org/en/latest/ (access date: 20.08.2013).

[5] Model–view–controller – Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Model-view-controller (access date: 20.09.2013).

[6] Sandhu "Role-based access control." *Advances in computers.* 46 (1998): 237-286.

[7] T.Berners-Lee, R. Cyganiak, et.al. On Integration Issues of Site-Specific APIs into the Web of Data. DERI Technical Report 2009-08-14. URL: http://linkeddata.deri.ie/sites/linkeddata.deri.ie/files/rw-wod-tr.pdf (access date: 20.09.2013).

[8] Virtuoso Open-Source Edition URL: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/ (access date: 30.05.2013).

[9] David MacKenzie, Paul Eggert, and Richard Stallman (1997). Comparing and Merging Files with GNU Diff and Patch. *Bristol: Network Theory*. ISBN 0-9541617-5-0.

[10] PIZZA Protege OWL tutorial at Manchester (School of Computer Science - The University of Manchester) URL: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/ (access date: 20.09.2013).

[11] SWI-Prolog's home. URL: http://www.swi-prolog.org/ (access date: 20.08.2013).

[12] The Protégé Ontology Editor and Knowledge Acquisition System. URL: http://protege.stanford.edu/ (access date: 20.08.2013).

[13] Semantic MediaWiki. URL: http://semantic-mediawiki.org/ (access date: 20.08.2013).

[14] N.Heino, S.Tramp, N.Heino, S.Auer. Managing Web Content using Linked Data Principles – Combining semantic structure with dynamic content syndication. *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual*. pp. 245 - 250. URL:http://svn.aksw.org/papers/2011/COMPSAC_lod2.eu/public.pdf (access date: 30.05.2013).

[15] Cherkashin E.A., Paramonov V.V., et al, Model Driven Architecture is a Complex System, E-Society Journal Research and Applications. Volume 2, Number 2, 2011, pp. 15-23.