A Semantic Markup Technique Based on Ontology Polysystem

Evgeny Cherkashin*†‡, Kristina Paskal†‡, Igor Bychkov*
*Institute of System Dynamics and Control Theory at SB RAS, Irkutsk, Lermontov str., 134, 664033, Russia
†National Research Irkutsk State Technical University, Irkutsk, Lermontov str., 83, 664074, Russia
‡Irkutsk State University, Irkutsk, Gagarina blvd., 20, 664003, Russia
eugeneai@icc.ru, bychkov@icc.ru

***Abstract*—The described investigation proposes an approach to overcome known problem of site developers' participation in the real development of Semantic Web content. The site integration in a way of the Semantic Web originally defined is too hard to be used in a regular site maintenance. This results in the necessity of development of site content management software on the base of knowledge acquisition systems, where developers and users have to play a role of an information source for a system-integrated decision-making engine inducing formal knowledge by means of data mining.**

The logical layer is generated on the base of analysis of changes introduced by user, which are analyzed and interpreted. The variant of the interpretation (e.g., error correction of a value or a new statement definition) is determined by means of user interview and other factors. The theoretical basis of the technique is based on a polysystem representation of ontologies describing the domain. The presentation is a hierarchically fibered structure of concepts and relations, which are mapped between fibers by means of interpretation relations.

I. Introduction

In 2001 Tim Berners-Lee proposed a blueprint [1] of web development that is aimed at the implementation of network services with reasonable integration of logical layer of the information presented. The information is to be marked up semantically and the program agents are to take advantage of the markup as the logical layer for the information consumption and processing. The blueprint is called Semantic Web (SW).

One of the main problems of SW is the fact, that the regular users of web resources are not fond of the complex technological aspects of SW. They are interested in their practical problems solutions. In order to involve users in SW content development these aspects must be completely hidden. This results in the necessity of development of document and site content management software exploited within SW as knowledge acquisition systems, where the user is to play a role of an information source for a system-integrated decision-making engine inducing formal knowledge on the base of data analysis.

To be more comprehensible consider a content of a legal document, which in most cases contains meaningful information on legal entities relationships that is usually passed to other documents in a derivative forms. This suggests an idea to develop a form of the logical layer representation in a reasonably detailed form, which can be rendered as a regular text pages by means of context-depended (in sense of a legal document) text templates. The SW approach could supplement the idea with data formats and technologies of their storage and processing. SW represents logical layer as a network graph of notions and relations between the notions. For example, individuals mentioned in a legal document relate to the document as "part-whole", unless to speak even more explicitly in a meaningful context.

At present most of the use cases of domain ontology models are refining search results. Automatic procedures of ontology extraction from the documents are based on crawling documents in a warehouse, running data mining procedures on text content and metadata attributes of the documents. Simple observation of human behavior in the process of a document preparation will result in the confidence that meaningful parts of the document, which are usually expressed with the logical layer, are located at the points of document changes. Hence, a programming system automating the document preparation should track the changes and analyze them in time to extract ontology structure, i.e., objects, concepts and relation between them.

The logical layer is induced during content edition by means of data mining and user interview. The context of the knowledge acquisition consists of a polysystem [2] of ontologies describing the domain, the source document (its text representation and current logical structure), its list of modifications in transaction, user's action history, and answers to the interview questions clarifying meaning of text content changes and properties of the relations constructed. As a result, new or modified triples `<subject, relation, object>` are constructed. Collected triple data, metadata and knowledge of the logical layer can also be crawled on a regular basis to grasp patterns and functional relations between attributes (triple's objects). In the last case a relational table could be induced to rise the efficiency of data storage and processing.

The present trends of internet information system development show that the systems become a web-services and are oriented to support social networks [3]. The data flow processing there in most cases is input, storage, filtering and transmitting data, i.e., the social networks integrate rather than aggregate data, e.g., to produce reports. Hence, the usage of special design techniques for the middle-ware layer of the software, such as object-oriented design, is not a significant technological advantage. Social network (SN) agent software are interconnected with standardized protocols and data structures, and have the same way of integration with other social networks.

Let us emphasize the properties of social network software:

- there are no predominant common global task to be solved by agents (computer and human) of the SN;
- each agent solves its special task acquiring data from other data sources and agents, hence, the agent's APIs and supported data format must be standardized;
- human users of the social network do most of the aggregation tasks personally including subconscious joint processing of unstructured and semantic information;
- a good practice is to use cheap virtual hosting to run the a SN node, therefore, the information system itself must be written according to the restrictions of a server-side programming language, like PHP, and utilize the computational and data storage resources of client workstation and its browser environment.

In this paper, we continue the development of the approach [4] of document and site content management and integration within a general study [5], and briefly consider the procedure of polysystem ontology usage in the process of user dialog maintenance.

II. Document Markup

Resource description framework (RDF) standard describes informational resources with triples `<subject, relation, object>` in a context. A set of triples forms a graph (network) of data and relations reflecting knowledge. It is convenient to divide graphs to subgraphs and construct hierarchic complexes [5], resulting a hierarchy of contexts. In a general case, a context affects the interpretation of the set of triples coupled with it. For example, in a legal notary document family name and passport data are presented as texts in different parts of the document, but they related to the single person in the context defined by the document. So the parts of the document form the contexts of text rendering for the passport data.

The rendering engine we considered in [4]. According to this approach, all the template data for document rendering are also stored in the ontology graph. We also represent the views and algorithms implementing controllers in the sense of MVC (Model View Controller) [6] pattern with triples. From a general point of view, the content of the document is reconstructed from a tree graph, where almost each node is both subject and object of their corresponding relations. The root node in a document can be only a subject, and the leaf nodes are objects only. The document root node becomes object node as being referred from document sets and hierarchies organized in a document warehouse.

III. The Outline of Markup Scheme

The proposed way of semantic document markup is to track and interpret the results of document editing. Let's suppose that a document body change affects the meaning of the document, hence, alters its logical layer. The following trivial modifications that are the elementary error correction are not considered to be a valuable information for the markup: a field value change (object of a triple), paragraph text editing that might imply its original template correction. The changes of our interest result in a new triple relation construction between context subjects and old/new object value. The content management system and a built-in editor play an active role in the markup process, and the user is a source of supplementary information.

The first question that user have to be asked is to determine weather a change is an above mentioned trivial one. If it is not, then the semantic layer enrichment procedure is carried on. The source of data to decision-making procedure of a triple formation/modification is an environment containing a) logic structure of the previous version of the document; b) the source version of the document content as text, c) a text modification expressed in diff format [7]; d) user answers to the question stated by the system, refining the semantics and structure of the modifications; e) user actions prior the modification done. The list of the previous user's actions can reveal a general intention, e.g., if user made a copy of document and changed an individual name, this would imply the necessity to fill in the passport data, which, in turn, implies filling in a number of triples with new values representing the new individual (subject) of the document.

At the next step of the markup process the extent of the modified value is determined. It could be a word, a phrase, a sentence, the whole object value, or an extent of a text markup (e.g. HTML tag value). A resulting variant is specified by user.

The simplest way of the triple formation procedure is to allow user to choose subject and relation from list of all relevant subjects mentioned in the document and their relations and construct a new triple `<subject,relation,old object>`. The list of subjects is constructed from a tree of all the subjects of the edited document and the edited object itself. After the first choice is made, a list of all available relation is constructed from all known relation of the chosen subject, its class and parent classes. User must choose one relation to form the triple. In the case, when the list contains no adequate relation, a new one could be defined. New relation is always a subclass of an already existing one, which is also chosen from the list. New relations defined by inexperienced users must be periodically analyzed by knowledge engineers to get rid of semantic inaccuracy, contradictions, redundancy to the equivalents, and, hence, be refactored.

If a value of a triple is removed, then the triple must be removed too. The situation is acceptable if the minimal structural and semantic completeness of all the subjects of the documents holds, otherwise either the user delete action is prohibited or the chain of recursive deletion of the subjects is initiated. To control this behavior each subject class have to be accompanied with a list of minimal valuable triples that define the basis of the subject meaning. 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.

Addition of a triple might result in extending the document with new subjects and relations. For example, let one construct a new tripartite agreement from an existing bilateral one. In the new agreement the third individual appears, so the addition of a new family name of the individual results in construction of a) subject for the individual, b) filling in the necessary triples, as

well as c) definition of a new relation between document and the third individual as a subclass of `structuralElement:` relation. This stage is necessary as all new subjects must be in a relation.

IV. Theoretical Generalization of the Acquisition Process

The above described approach is a simplified step-wise implementation of the procedure of polysystem analysis and synthesis [2] (PPAS). This is a general procedure for system analysis of domains. The essence of the approach is that domain and any its object or process can be fibrated and represented in multidimensional space of fiber-coordinates. This approach is used in decomposition, analysis and synthesis of complex systems.

The fibers consist of concepts and morphisms between them. For each concepts of a fiber there exists another concept in all the fibers, which is identical to the former concept via an interpretation (e.g., substitution mapping). All the fibers are element-wise mapped to the abstract system fibers. The same is true for the morphisms. In comparison to the widely known system approach, where the principle of interconnection of the elements is fundamental, polysystem analysis assumes the hypothesis of fibration, i.e., it supposes a possibility of representation of the object under investigation as disjoint, thus, mutually unbound fibers (subdomains). So, in a generalized insight a polysystem is a system itself, and polysystem analysis is a new form of system analysis.

Now then, according to PPAS [2] each concept of an aspect should be presented via interpretation morphisms in all other fibers as well as its relationships with adjacent concepts. For each of the fibers, a complete theory of the domain can be constructed. Each theory differs by their fundamental concept, but it is similar to all other via concept interpretation. This allows us to induce new axiomatic theories of domains in the image and likeness of the known ones. Each system domain (fiber) of knowledge fibrates multiply and sequentially, giving raise of the polysystem of representation of an object under investigation. All the system theories are combined in the unified model describing the domain and the object under investigation.

The application of the PPAS to the process of data and knowledge acquisition in the documents expressed as simultaneus polysystem fibering the domain in various coordinate systems (aspects) such as follows: structural organization of a document (mereological aspect), expressing domain of activity area, hierarchical inheritance of various kind of documents (domain system concept inheritance), people and organization relationship, deontic logic representation of the document meaning, etc. Having established new class or new relation between two classes, the concept or relation should be reflected as a substituted (interpreted with) one in other ontologies (e.g., mereological). To make the polysystem to be easily understandable to users, definitions for the concepts and relations must be supplemented in a natural language using terms of those ontologies. The traditional requirement of affiliation of an object to a class, and inheritance between classes and relations reflects the idea of hierarchical polysystem fibering. The relations of inheritance and affiliation are the examples of concept interpretation between the fibers.

The conceptual methodological basis of PPAS allows us to a) control the completeness of the system of objects in a document with respect to their interpretation to a predefined abstract system fiber, which we roughly call “template fiber”; b) verify the soundness with respect to the semantics of the relation defined again via interpretations; and in a distant future c) automatically produce program implementing the fiber theories via interpretation and combination of the existing algorithms of other fibers.

V. Maintaining a Dialog With User

The through interpretation feature of the polysystem layers could be used to maintain user interview dialog on the base of an inference by analogy represented as movement along relations, morphisms and interpretation in a polysystem of ontologies.

In order to realize the dialog procedure, there should already exist system fibers representing “part-whole” relations, class hierarchies of the abstract concepts, correspondence of objects to classes. The aspects of the domain is represented by set theories and necessary domain layers, e.g., the fiber defining that “man” and “woman” are disjoint concepts. The user dialog is used to determine that the object belongs to a context (fiber), recognize its interpretations into concepts of other fibers, and to provide all the context to be complete and sound with respect to their template fibers. The dialog is maintained on the base of analysis of known properties and restrictions (relations) of the subject: type of data representing the subject, its location in the context hierarchy, existing relations with subjects and objects, etc.

Let’s consider simple family relationships in fig. 1, where most of the obvious interpretation relations are hidden to make the figure more readable. There showed a part of the domain polysystem consisting of four main layers, which model various aspects of Bob’s family. Layer 1 represents the family itself, it is the only layer having objects as concepts; layer 2 represent main roles of individuals in a family; layer 3 shows genders as two opposites; and later 4 corresponds to a rough mereological model “part-whole”.

Let’s construct a sequence of questions to determine a role of Jul in Bob’s family. It is supposed that we know only that Jul is a woman, and she has been mentioned in a document related to Bob’s family. To put Jul in layer 1, in a edited context of a document, we must ask a general question: “Is Jul a part of Bob’s family?” The question is constructed according to a following intuitive inference. All members of a family are “partOf:” the family. This fact is shown as an arrow from relation “hasFather:” of Layer 2 to “partOf:” of Layer 4, other four arrows are hidden. So, in order to determine the fact that Jul is a member of Bob’s family, it is sufficient to ask “Is Jul partOf: Bob’s-family?”. The identifier of the example family is devised from the tradition to name families after their master of housekeeping (MOH) person, that is denoted by arrow from individual Bob (layer 1) to concept MOH (layer 2). Note, that layer 1 interpreted by layer 2 by means of a functor of theory of categories.

If user answered positively, i.e., Jul is a part of the family, then we must refine her role further, as role “partOf:” is not allowed in layer 1, and, hence, layer 1 is not complete yet

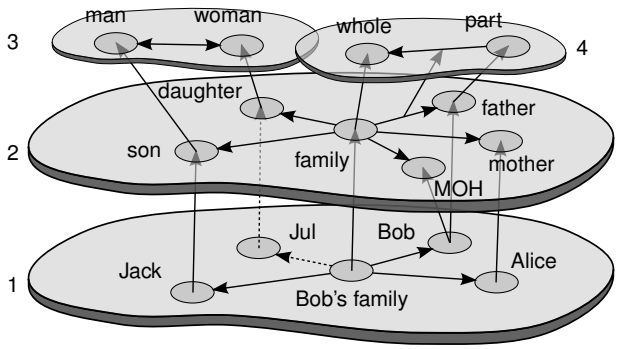

Figure 1. A Usage Example of an Ontology Polysystem

with respect to template layer 2. Jul is a female by definition, so she cannot be father and son. If she is, she must be male, but it is impossible as male and female are formally opposite concepts. If multiplicity of relation "hasMother:" in the layer are defined as one, then we will also know, that the family already has mother. The only possible choice is Jul is daughter, resulting in another general question:"Is Jul a daughter?" If the answer to the question is also positive, finally a new triple constructed `<Bob's-family, hasDaughter:, Jul>`. Now the layer of Bob's family is complete with respect to the template layer 2.

If, e.g., Jul's gender is unknown, then the second question is constructed as a special question "What role does Jul have in Bob's family?" and a list of two possible answers is shown: "hasDaughter:" and "hasSon:". Another variant of the question can be synthesized if only two alternatives exist: "Which of the following two roles does Jul play in Bob's-family: hasDaughter: or hasSon: ?".

In fact in the original example the question about a role of Jul is redundant as there exist only one role for a woman being a part of Bob's family. If we remove mother Alice from the family then the question have to be asked, even for incomplete families. The general questions are asked when we could narrow a list of alternatives in further questions. The queue of the questions being asked could be defined at runtime by analogy to decision tree construction procedure, e.g., by analyzing entropy gain [8].

VI. A Notary Office Usage of the Technique

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 in a balance reasonable for our study. For example, formalized data are the passport data of individuals, registration data of vehicles. This kind of data are normally stored in rational databases, passed from one document to other documents mostly structured and unchanged, raising a bundles of related documents and patterns of document flows. The final document content structure significantly depends on entry fields data filled in, e.g., form of notary signature field depends on legal capacities of the individuals mentioned in the main document body. Unformalized data are various enumerations of legal empowerments of the individuals, article codes, and explanatory text.

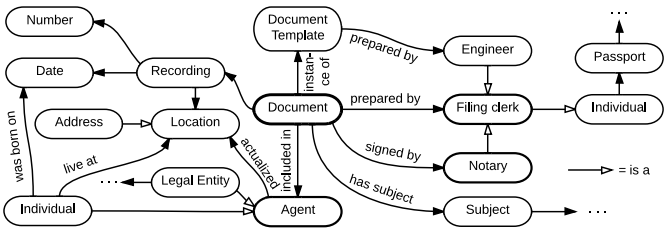

Figure 2. Upper Layer of Notary Office Ontology

At present most of notary legal documents are generated from templates. In experimental office, the document templates are prepared by engineer in collaboration with a secretary having necessary experience of use of the existing software. Templates are HTML forms dispersed in the text of document. Logical structure of a document is defined by field names, and the procedures of information processing are implemented as simple form processing routines. The output document is rendered by means of value substitution instead of fields. The field names are defined by identifiers according to a special rules so that fields implicitly combine into a subject. This simple formalization allows one to carry on basic transformation operations on roles of the individuals by means of renaming or exchanging values of the field sets. Application of semantic approach to the logical layer representation of the notary document content is aimed at involvement of the secretaries (office stuff with low engineering skills) into development of templates and constructing hierarchical arrangements of documents.

In new approach, 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 personal data, e.g., for partially capable children. As we just noted, all the individuals are in explicit relation to the document. The triples `<letatt998 containsPrincipal: indiv_78>` (commas are omitted) and `<letatt998 containsProxy: indiv_79>` could define a principal and proxy of the document. The mentioned relations are specifications of abstract `hasIndividual:` and `structuralElement:` relations. This polysystem 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 role swap function is implemented as user interface widget generated as a result of recognition of certain subject-to-subject relations indistinguishable in a fiber, or specific patterns of relations in the document body. A general graph structure of the abstract level of a notary office automation ontology is drawn in fig. 2.

VII. The Software

The software developed within the project will include two major versions. The first one will be a client and server combination, where client will drive the data mining and visualization procedures, and server will control storage and

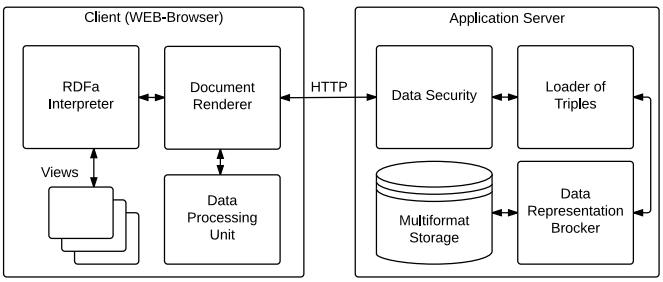

Figure 3. Program System Architecture

access rights to a graph of the domain. The second version of the software is a desktop application that is a fusion of the client and server functionality. The application is intended for corporate and personal off-line usage. The following are the requirements of the software to be fulfilled in our investigation:

1) the software should be developed as an open-source project, and we hope to involve in our project a free community of engineers and users;
2) server software must be not too consumptive of computational and memory resources of a shared hosting service;
3) in accordance to 2, a support of various database back-ends are to be implemented to store triple data.

We orient our efforts of implementation to a popular hosting environments that include PHP as programming language and operational environment, MySQL- and PostgreSQL-servers as a storage, functioning under Linux operation system. As a primary client, Firefox web browser has been chosen, and its JavaScript implementation is being considered as a client programming language.

Architecture of the first variant of the software is presented in fig. 3. Server side deals with data storage and security. “Multiformat Storage” module supports various techniques of data representation, it could be a SQL-server. The purpose of “Data Representation Broker” module is to convert triples in a format that is better support special tasks, such as aggregations, and pass the conversion result to the storage. In the backward direction, the broker restores triples from the storage as RDF entities. 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 security is supplied by Data Security module.

At client side the triples are rendered as HTML texts and combined into a document by “Document Renderer” module. The result is either interpreted further by “RDFa interpreter” or processed in “Data Processing Unit”. New concepts and relations are sent back to server as RDF- and RDFa-content.

A. Implementation Details

Representation of domain graph and PHP’s access to graph components is implemented on the base of EasyRdf library. The library supports loading RDF-files and represent them as PHP’s objects. The objects can be accessed in a usual object-oriented way and by means of the special standard query language SPARQL.

Storing RDF-triples in MySQL is to be implemented with ARC2 library, which also supports SPARQL-queries. ARC2 is a development platform with SW functionality, allowing one to extend the standard set of routines by means of plug-in modules as well as to create trigger functions fired as soon as SAPRQL query finished.

Text composition library Zope Page Template (ZPT) [9] is used as a generic text rendering engine out of the logical layer. ZPT is cross-platform and allows representing templates and rendering the triples inside. The templates also stored in RDF. ZPT technology and its descendants, like Chameleon [10], allow programmers and designers to share their common work on a dynamic HTML page because of ZPT is based on a declarative approach to template definition.

ZPT engine includes interpreters of several query expression languages to a logical structures of model and template: TAL (Template Attribute Language) and METAL (Meta TAL). There are also tools for internationalization (I18N) support. Developers of ZPT successfully solved the following standard web problems:

- separation application logic and user interface;
- utilization of existing HTML-page design software and editors (e.g., Dreanweaer) for manipulation of template content with conserving TAL and METAL expression;
- trivial cycle programming structures are replaced with a declarative constructions;
- thanks to METAL templates are manipulated in a powerful manner — cutting and pasting subtrees of a HTML page — using triple or object data.

To date there are many implementations of TAL and METAL for various programming systems, including PHP (PHPTAL). The same templates can be used in all of them if programmer uses only the basis syntax and semantics of TAL. For client-side language JavaScript, ZPT has been implemented by DisTAL library, which can also dynamically load model data into rendered template, resulting consumption more CPU resources of client workstation and less of server.

The storage module of desktop application should not be too resources consuming, that’s why we propose usage of in-process database libraries which are also do not require special configuration and engineer’s periodical maintenance. One of the outstanding in-process database engine is Kyoto Cabinet library [11], which is high-performance implementation of BerkeleyDB standard. The library maps blocks of bytes, i.e., a key to a value, and the mapping is realized by means of B+-trees and hash tables. Kyoto Cabinet supports protection against data loss even due to partial physical damage of hard disks. The protection is achieved by special file format and data replication among a cluster of workstations.

Thus, the nowadays open-source software of internet application development and relatively cheap server computational resources completely cover all the requirements of the software under development.

VIII. Similar Projects

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 [12].

In comparison to Semantic MediaWiki the project OntoWiki [13] 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.

IX. Conclusion

The research is aimed at support of Semantic Web with algorithms and software automating text document markup and its logical model induction. The approach is based on polysystem representation of a domain and a document content (e.g., web site) as an evolving ontology. The usage of polysystem of ontologies and the procedure of polysystem decomposition [2] allows us to solve a number problems in the activity related to knowledge acquisition in the domain. New knowledge are represented as fibers of categories with property of through cross-fiber interpretation. In the future, the possibility will arise for automatic construction of algorithms and programs implementing a fiber theory in the image and likeness of an already constructed ones. At present the polysystem of ontologies allows us to rate and categorize facts on the domain by analogy with the already known structures, as well as obtain new interpretations of new recognized concepts.

The main part of the paper is devoted to a general description of a technique of maintenance of an interactive process of logic model induction of document content on the base of data mining of changes of various versions of the document. A problem solution outline for user dialog conduction aimed at modification and extension of the logical model with new objects and relations has been considered. The dialog is maintained by analyzing the completeness of the constructed model and logical inference in the polysystem representation of the domain ontology. A pilot software architecture has been presented as well as brief overview of used modules and libraries, their properties and features have been assessed.

As a testing ground for the technologies under development we decided to automate a notarial office, where the legal document representations has both well formalized and unformalized text fragments. On the base of the technology a social network of document data exchange can be devised. The security of the document transfer can be provided as off-line data streams: each physical document is accompanied with its bar or QR-code encoding the RDF-data of the document.

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

[2] A. K. Cherkashin. “Polysystem Analysis and Synthesis. Application in Geography,” Novosibirsk, Russia “Nauka” Publ. co. 1997. – 502 p. (In Russian)

[3] Social network – Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Social_network (access date: 09.01.2015).

[4] E. Cherkashin, P. Belykh, D. Annenkov, K. Paskal, “A Document Content Logical Layer Induction on the Base of Ontologies and Processing Changes,” Procs. of International Conference on Applied Internet and Information Technologies, October 25, 2013, University of Novi Sad, Technical Faculty “Mihajlo Pupin”, Zrenjanin, Republic of Serbia, pp. 252–257. URL: http://www.tfzr.uns.ac.rs/aiit/archives/AIIT2013/Proceedings.pdf.

[5] E. Cherkashin, V. Paramonov, et al, “Model Driven Architecture is a Complex System,” E-Society Journal Research and Applications. Volume 2, Number 2, 2011, pp. 15–23. URL: http://www.tfzr.uns.ac.rs/esociety/issues/eSocietyVol2No2.pdf (access date: 09.01.2015).

[6] Model–view–controller — Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Model-view-controller (access date: 09.01.2015).

[7] D. MacKenzie, P. Eggert, R. Stallman. “Comparing and Merging Files,” URL: http://www.gnu.org/software/diffutils/manual/diffutils.pdf (access date: 09.01.2015).

[8] Information gain in decision trees — Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Information_gain_in_decision_trees

[9] “Zope Page Templates Reference,” The Zope2 Book. URL: http://docs.zope.org/zope2/zope2book/index.html (access date: 09.01.2015).

[10] Chameleon – Chameleon 2.10 documentation. URL: http://chameleon.readthedocs.org/en/latest/ (access date: 09.01.2015).

[11] Kyoto Cabinet. URL: http://fallabs.com/kyotocabinet/ (access date: 09.01.2015).

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

[13] 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).