Towards a standard protocol for community-driven organizations of knowledge

Chao Zhou, Christophe Lejeune, Aurélien Bénel
Laboratoire Tech-CICO, Institut Charles Delaunay (FRE CNRS)
Université de Technologie de Troyes

Abstract. This paper deals with the "Web 2.0", where every user can contribute to the content, "harnessing collective intelligence". After studying what makes the success of services like *Google Base*, *Del.icio.us* and the *Open Directory Project*, we propose a unifying "REST" protocol for this kind of community-driven organizations of knowledge. The aim is to make the collaboration possible beyond the boundaries of the software and of the resulting communities.

Keywords. Web 2.0, Communities, Knowledge Management, REST Web services.

Introduction

The future of the Web was planned to be a *Semantic Web* with "content that is meaningful to computers" **[2]**. What we got instead is a *Web 2.0*, where every user can contribute to the content, "harnessing collective intelligence" **[14]**.

One should note that the move is not only from a formal semantics to a social one, but also from an innovation process lead by a consortium to a new one lead by independent socio-economic actors. The drawback of such a process is the resulting "babelization" between the different software services which makes it difficult to collaborate between the different user communities.

In the following pages, we will focus on community-driven organizations of knowledge. After studying what makes the success of *Google Base*, *Del.icio.us*, and the *Open Directory Project*, we will propose a unifying infrastructure for this kind of services.

1. Success stories

1.1. Google base

Google Base[1] is a "beta" service by Google which allows anybody to:
- Create an item of any type and describe it,
- Look for items satisfying to criteria.

This service is intended to become *the* worldwide database for any type of items (even scientific ones like genes). For now, it is mainly used for classified ads (dating, housing, used cars…).

In fact, the data structure of *Google Base* reminds the one of the old *Machine Readable Catalogue* (*MARC*, still in use in public libraries) in the way it is "schema neutral". As *MARC* came in different "flavors" (*LCMARC*, *UKMARC, UNIMARC*…) chosen by librarians to fit their books and patron needs **[10]**, Google Base allows the user to use any attribute names (existing or new ones) to fit her item type and her needs (cf. Fig.1).

Nevertheless, the "report bad item" feature could indicate an interesting gap between the social process involved in *Google Base* and the library sciences goal of objectivity. That could be the reason why this universal database is in fact used only when the user *owns* the item (cf. Fig.1) and therefore is the only person who can describe it.

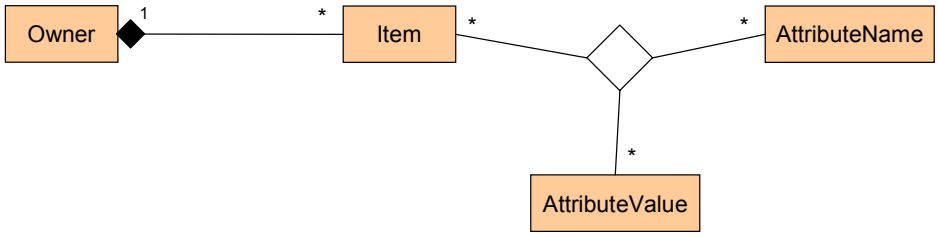

Fig.1 – Reverse engineering of *Google Base* (UML class diagram)

1.2. Del.icio.us

Del.icio.us[2], a service first created as a hobby and now owned by *Yahoo!*, allows anyone to:
- Keep a bookmark of a web item and describe it with free keywords (called "tags"),
- Share them with other users,
- Discover new items by browsing popular and related tags,
- Make one's own description to existing items.

Del.icio.us aims at creating a directory of web pages by putting together a bunch of personal bookmarks. In order to gain objectivity, they had chosen a democracy-like model where every opinion can be expressed but is considered to be significant only when it is shared by a lot of people.

---

[1] http://base.google.com/
[2] http://del.icio.us/

In a way, the data structure of *Del.icio.us* (and other similar "folksonomies" like *Flickr*) reminds the one from *Xanadu*: "the original hypertext project" **[13]** in which users were able to reuse fragments and links in different "documents". But the difference between a "document" and a "tag" is that a tag is not *owned* by a user. The tag is collective and therefore only the statement saying that a document is described by a tag is attributed to a user (cf. Fig.2). But, are the tags "mydog" and "todo" really collective [11]? Do these tags even mean something in a shared place? In the same way, does "apple" mean the same thing for geeks, cookers and New Yorkers? To be really collective, tags should be defined inside a "viewpoint": a *language* used by a community. Then it would not be the same "apple" tag, just as "pain" is not the same in English and in French.

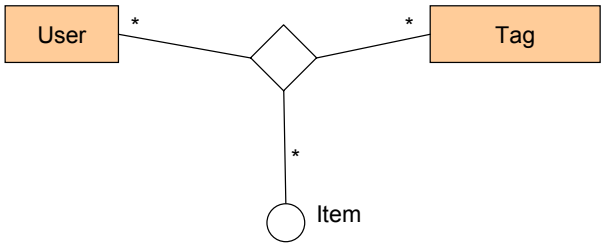

Fig.3 – Reverse engineering of *Del.icio.us* (UML class diagram)

1.3. The Open Directory Project

As all directories, the *Open Directory Project*[1] is a tool to help users locating information on the Internet. The website home page of the project proposes some general topics as a starting point of the query. Crawling from general topics towards more and more specialized rubrics, the user can specify her query so that (and up to) she will find a list of websites containing the information he is searching.

A directory is thus a hierarchical structure of categories. The main top category contains all the others (it stands as the front menu of the home page). This menu features a dozen of general topics (such as arts, sciences or health). Each of these general categories contains a branch of imbricated sub-categories and websites references.

Leaving the user's point of view for the designer's one, the *Open Directory Project* is a community of volunteer editors. Each editor maintains (at least) one category (which means that editors are thematically skilled). She is responsible for all the content of this category. This includes recording websites, describing the category, inserting sideways links, and managing all subsequent subcategories. The core business of the editor is to provide the directory with website references. This includes, for each recorded web sites, to insert an address (URL), a title and a short text describing its content. Editors also redact the description for the rubric they are in charge and insert sideways links from category (these links are described hereunder).

The fact that they can manage subcategories means that scopes of action differ from one person to another. This situation leads to propose that the thematic tree is coupled by a social tree (with social issue in selected cases).

[1] http://www.dmoz.org/

The aim of the community (as a whole) is to propose an alternative to other information retrieval tools. The ultimate purpose is to provide users with a tool giving more adequate results than search engines. The very idea resides in that:

- The path of the categories situates and contextualizes the information (contrary to keywords query introduced in the input field of a search engine front page),
- Each reference is described by an expert.

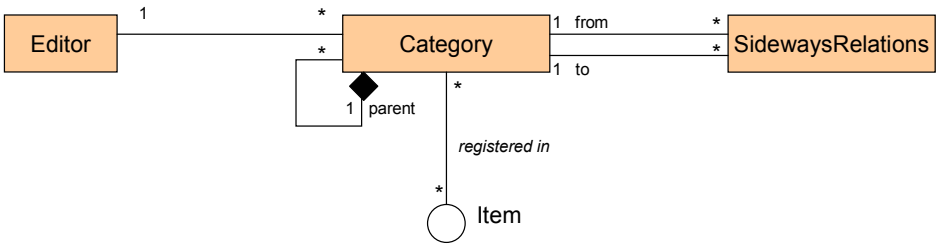

Fig.2 – Reverse engineering of the *Open Directory project* (UML class diagram)

Founders of directories face two problems. The first is related to human resources, the second with tree structure. Given that directories databases are constructed by humans (contrary to search engines that are computer-processed), these projects need large teams of skilled specialists. This core weakness is solved in the case of the *Open Directory Project*, thanks to its organization in a community of benevolent contributors (as free software programmers).

The second problem is the tree structure of the database front end. Even if this shape organizes the information, it can yield to locate some rubrics that concern close topics in different branches. This raises usability problems. Used to this question (through similar tools as thesauruses), information scientists solve the issue with sideways relations that allow the user to glance through the database. Known as "related terms", these horizontal bridges are indicated by the "see also" heading **[1]**. Directories feature also such relations. In the *Open Directory Project*, they are of three kinds: related categories, alternate language and symbolic links.

As signaled by the name, related categories implement a relation similar to the "see also" link from thesauruses. The complete path and name of the target categories are featured and the target (sister) category is considered to cover a close theme to the origin category. The relation can be reciprocal but this is not necessary.

Alternate language links relate categories that cover the same theme in different language. This means that the global tree of the *Open Directory Project* includes duplicate hierarchies in each language. This division was not in the original model but was introduced when more and more non-English members join the project. At a first stage, language branches were created and relations between them were indicated with the related categories features. Then (late 2000), the alternate language link was introduced. The link only indicates the name of the language to which the target category belongs (only one equivalent by language is permitted).

The symbolic links are sideways relations that include a peculiar semantics. The target category can be considered as a subcategory (a child) of the origin category. Symbolic links are listed among other effective sub-rubrics and are signaled and distinguished from the latter by a trailing "at sign" (@). Neither the path, nor the name of the target category is featured; the name of the symbolic link is chosen by the indexer.

2. Towards a unifying infrastructure

The following section describes the *HyperTopic* protocol, named from the underlying data model: the *HyperTopic* Model **[6]**. Our goal is to propose this protocol as a standard for services aiming at community-driven organizations of knowledge.

The protocol is designed in a "REST" style to achieve visibility, reliability and scalability. REST is an acronym standing for "Representational State Transfer". REST is not a standard; it is an architectural style for distributed network systems **[9]**. The motivation of REST was to find out which characteristics made the web successful, and use these characteristics to guide the evolution of the web **[7]**. An important rule in REST is that every resource should have one URI. The components in the distributed system could use a set of HTTP methods (POST, PUT, GET, and DELETE) to manipulate those resources. Representations in REST style protocol usually are HTML or XML files that contain information and links to other resources. The components of the distributed system can navigate from one state of representation to another state, simply by following the links.

2.1. Objects URI

One of the most important characteristics of REST is about exposing resources through URIs **[12]**. There are different types of object in the *HyperTopic* model (Fig.4): Actor, Viewpoint, Topic, Entity, and Attribute. All of those should be uniquely addressable through URIs. A client could realize representational state transfer from one object to another object (e.g. from viewpoint to topics, or from topics to entities) by following those URI.

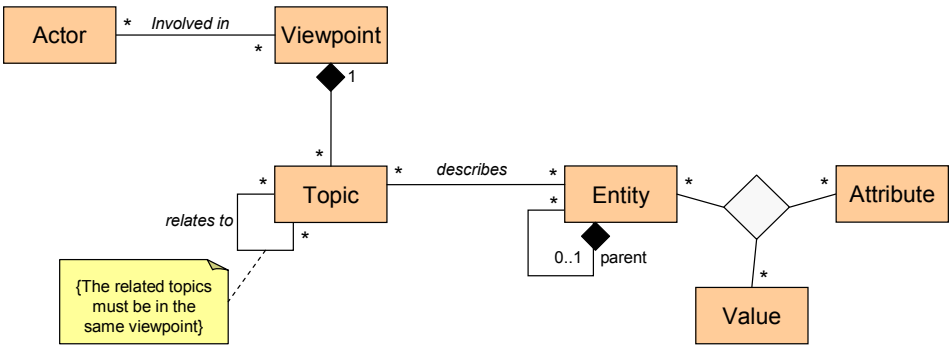

Fig.4 – The *HyperTopic* Model (UML class diagram)

Actor

The actors involved in reading and writing viewpoints are identified by a login. Assuming there were on www.example.org an actor whose login is 'linuxfans', its URI would be: *http://www.example.org/actor/linuxfans/*

Viewpoint

According to user's roles, a user could visit or manage one or several viewpoints by their URI. Assuming there were on www.example.org a viewpoint which ID is 1, its URI would be: *http://www.example.org/viewpoint/1/*

Topic

Every topic must belong to one and only one viewpoint. The URI of that topic will contain the viewpoint identifier and use hierarchical structure to represent the relationship between viewpoint and topic. The URI of topic #2 from viewpoint #1 would be: *http://www.example.org/viewpoint/1/topic/2*

Entity

Every entity should have one URI. An entity is defined by a persistent name. For example, the URI of the "Amaya" software entity would be:
http://www.example.org/entity/AMAYA/

Attribute value

An attribute value is identified by its name and its value. For example, the URI of 'INRIA' as an 'author' is: *http://www.example.org/attribute/author/INRIA/*

2.2. XML structure

In the *HyperTopic* protocol, software and service transfer data in XML streams. The following tables describe the *HyperTopic* document format. Those tables use standard XPath notation, slashes to show the element hierarchy, and an "at sign" indicates the attribute of an element.

Actor

XPath	**Cardinality**	**Description**
/actor/@name	Optional	Actor Name.
/viewpoint	*	Viewpoints which this actor could visit or manipulate
/viewpoint/@xlink:href	Required	Viewpoint URI

Viewpoint

XPath	**Cardinality**	**Description**
/viewpoint/@name	Optional	Viewpoint Name.
/viewpoint/actor	*	Actor which could visit or manipulate this viewpoint
/viewpoint/actor/@xlink:href	Required	Actor URI
/viewpoint/topic	*	Topics which are linked to the viewpoint.
/viewpoint/topic/@xlink:href	Required	Topic URI.

Topic

XPath	Cardinality	Description
/topic/@name	Optional	Topic Name.
/topic/viewpoint	1	Viewpoint which the topic belongs to.
/topic/viewpoint/@xlink:href	Required	Viewpoint URI.
/topic/relatedTopic	*	Topics linked to the current topic Note: The related topics should be in the same viewpoint.
/topic/relatedTopic/@relationType	Optional	The relation type between the current topic and the related topic.
/topic/relatedTopic/@xlink:href	Required	Related topic URI.
/topic/relatedTopic/@status	Optional	Status of related topic (active or inactive).
/topic/entity	*	Entities described by the topic.
/topic/entity/@xlink:href	Required	Entity URI.
/topic/entity/@status	Optional	Status of the link to the entity (active or inactive).

Entity

XPath	Cardinality	Description
/entity/attributeValue	*	Attribute values which belong to the Entity.
/entity/attributeValue/@xlink:href	Required	Attribute value URI.
/entity/attributeValue/@status	Optional	Status of the link to the attribute value (active or inactive).
/entity/topic	*	Topics which describes the Entity.
/entity/topic/@xlink:href	Required	Topic URI.
/entity/topic/@status	Optional	Status of the link to the topic (active or inactive).

Attribute Value

XPath	Cardinality	Description
/attributeValue/entity	*	Entities described by the attribute value.
/attributeValue/entity/@xlink:href	Required	Entity URI.
/attributeValue/entity/@status	Optional	Status of the link to the attribute value (active or inactive).

2.3. HTTP Methods and Status Codes

In a RESTful protocol, a client can use GET method to retrieve resources. Use POST method to create a new resource with new URI. For example, a client can send POST request with XML payload to URI "/viewpoint/" to create a new viewpoint. After the successful execution, the server will return a newly created URI with status code 201.

The PUT method is used to create a new resource or to replace an existing resource with URI. If the request-URI refers to an existing resource, the server will replace the resource with the enclosed resource. If the request-URI does not point to an existing resource, the server would create the resource with that URI. In the *HyperTopic* protocol, to trace the changes, the DELETE method does not actually delete the resource but just inactivate it.

The following table describes what the status codes mean in HTTP.

Code	**Details**
200 OK	The request has succeeded.
201 CREATED	The request has been fulfilled and resulted in a new resource being created.
205 RESET CONTENT	Modification of a resource has succeeded.
400 BAD REQUEST	The request could not be understood by the server due to malformed syntax.
403 FORBIDDEN	The server understood the request, but is refusing to fulfill it.
404 NOT FOUND	The server has not found anything matching the Request-URI.
500 INTERNAL SERVER ERROR	The server encountered an unexpected condition which prevented it from fulfilling the request.

The following table gives the typical status codes in the *HyperTopic* protocol that could be returned.

Actor

	Description	**200**	**201**	**205**	**400**	**403**	**404**	**500**
GET	To get actor							
PUT	To create or update actor.							

Viewpoint

	Description	**200**	**201**	**205**	**400**	**403**	**404**	**500**
GET	To get viewpoint list. To get viewpoint							
POST	To create viewpoint.							
PUT	To update viewpoint.							
DELETE	To trace changes, we do not really delete the viewpoint, and instead we just inactivate it.							

Topic

	Description	**200**	**201**	**205**	**400**	**403**	**404**	**500**
GET	To get topic.							
POST	To create topic. To update topic.							
DELETE	To trace changes, we do not really delete the topic; instead we just inactivate it.							

Entity

	Description	200	201	205	400	403	404	500
GET	To get entity.							
PUT	To create entity or update entity.							

Attribute Value

	Description	200	201	205	400	403	404	500
GET	To query attribute value							
PUT	To add new attribute value or update attribute value.							

2.4. Other resources

Although a client can enter the framework by actors URI, it could also do it by other special resources URIs. Those URIs will be used through GET methods only.

A client could use the following URIs to know the existing viewpoints: *http://www.example.org/viewpoint/*

The following table shows the format of viewpoint list:

XPath	Cardinality	Description
/viewpoints/viewpoint	*	Viewpoint
/viewpoints/viewpoint/@xlink:href	Required	Viewpoint URI

To know the existing attribute names, a client would use the following URI: *http://www.example.org/attribute/*

The following table shows the format of attribute list:

XPath	Cardinality	Description
/setOf/attributeName	*	Attribute name.
/setOf/attributeName/@xlink:href	Required	Attribute name URI.

A client would use the following URI in order to know the existing values for an attribute name: *http://www.example.org/attribute/author/*

The following table shows the format of attribute list:

XPath	Cardinality	Description
/attributeName/value	*	Attribute values.
/attributeName/value/@xlink:href	Required	Attribute URI.

A client could query an attribute. The query URI consists of an attribute URI and a SQL-like clause. For example: in order to find out the entities whose "type" is "Software", the URI would be:
http://www.example.org/attribute/type/upper(value)='SOFTWARE'

XPath	Cardinality	Description
/setOf/attributeValue	*	Attribute values corresponding to the clause.
/setOf/attributeValue/@xlink:href	Required	Attribute value URI.
/setOf/attributeValue/entity	*	Entities described by this attribute value.
/setOf/attributeValue/entity/@xml:href	Required	Entity URI.

Conclusions and future works

This paper challenges the future of the Web, to be related with socially (rather than ontologically) organized knowledge. We started with the description of three collaborative actual projects. Even though composed of volunteer lay people, these three communities exhibit peculiarities in the way knowledge is organized inside them. Our challenge was to propose a unified infrastructure for these. Based on the *HyperTopic* data model, we introduced the *HyperTopic* protocol to reach this aim.

After having implemented the server side of the project in *Argos,* we are currently working on the client side. This work is achieved by modifying two existing software: *Agorae* **[5]** and *Porphyry* **[4]**. The first one is a virtual marketplace allowing users to propose and describe material and immaterial goods (Fig. 5). The later is a digital library where scholars can publish and interpret differently document corpora (Fig. 6). Both of these software systems have been used by experts sharing their knowledge in collaborative project. Each user can construct her own viewpoint. A viewpoint can also be managed by groups of agreed individuals. This assumes that exposing concurrent view could fruitfully regulate and federate the community **[15]**.

Once *Agorae* and *Porphyry* are adapted to conform to the *HyperTopic* protocol, we are willing to propose this protocol as a standard draft to a normalization organization.

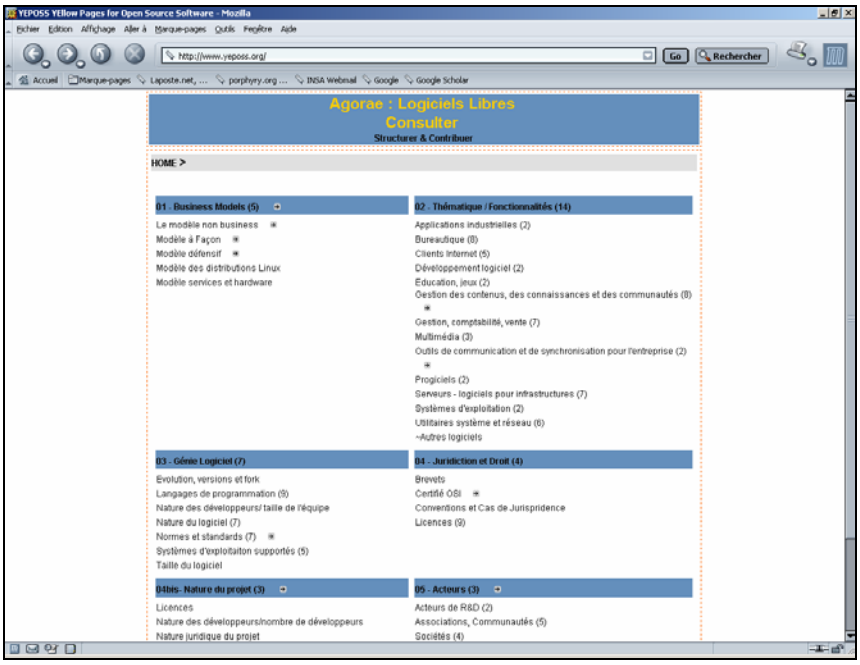

Fig.5 – "YEPOSS: Yellow pages for open source software" **[6]** (*Agorae* screenshot)

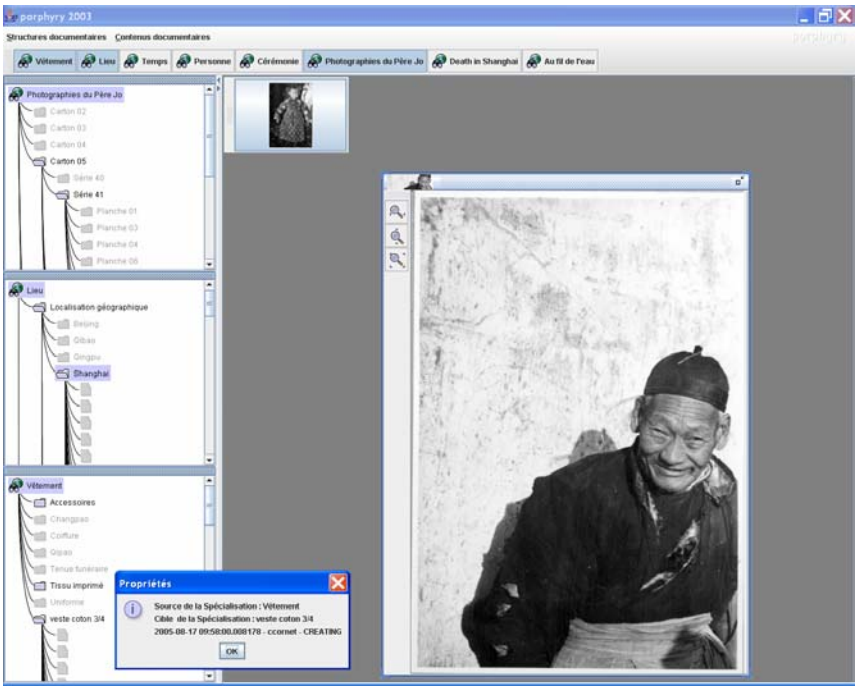

Fig.6 – "China in the 30s: Making history from photographs" by C. Henriot and C. Cornet from the "Institut d'Asie orientale" (*Porphyry* screenshot)

References

[1] Aitchison J., Gilchrist A., Bawden D., *Thesaurus Construction and Use: A Practical Manual*, Chicago : Fitzroy Dearborn Publishers, 2000.

[2] Berners-Lee T., Hendler J., Lassila O., The Semantic Web, *Scientific American, May 2001*. Available on: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF2>

[3] Berners-Lee T., Fielding R., Masinter L., *Uniform Resource Identifier (URI): Generic Syntax*, RFC 3986, The Internet Society, January 2005. Available on: <http://www.ietf.org/rfc/rfc3986.txt>

[4] Bénel A., Calabretto S., Iacovella A., Pinon J.-M. Porphyry 2001: Semantics for scholarly publications retrieval, *Proceedings of the thirteenth International Symposium on Methodologies for Intelligent Systems, Lecture Notes in Artificial Intelligence #2366*. Berlin: Springer-Verlag, 2002. p.351-361. Available on: <http://www.porphyry.org/Members/abenel/benel_ISMIS_02.pdf>

[5] Cahier J.-P., Zacklad M., Towards a Knowledge-Based Marketplace model for cooperation between agents. *Proceedings of COOP'2002 Conference,* Amsterdam: IOS Press, 2002. Available on Internet: <http://www.sociosemanticweb.org/jean-pierre/Coop2002.pdf>

[6] Cahier J.-P. Zaher L'H.. Leboeuf J.-P., Pétard X., Guittard C., Experimentation of a socially constructed "Topic Map" by the OSS community, *Proceedings of the IJCAI-05 workshop on Knowledge Management and Ontology Management*, Edinburgh, August 1, 2005. Available on: <http://www.limsi.fr/~xpetard/articles/cah-et-al-OM2005-Final.pdf>

[7] Costello R. L., *Building Web Services the REST Way*. Available on: <http://www.xfront.com/REST-Web-Services.html>

[8] Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P., Berners-Lee T., Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, The Internet Society, June 1999. Available on: <http://www.ietf.org/rfc/rfc2616.txt>

[9] Fielding R., *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, Irvine, 2002. Available on: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

[10] Furie B., *Understanding MARC Bibliographic: Machine-Readable Cataloging*, Washington: Library of Congress, 1998. Available on: <http://www.loc.gov/marc/umb/>

[11] Guy M., Tonkin E., Folksonomies: Tidying up Tags?, *D-Lib Magazine, Volume 12 Number 1, January 2006*. Available on: <http://www.dlib.org/dlib/january06/guy/01guy.html>

[12] Gregorio J., *How to Create a REST Protocol*. Available on: <http://www.xml.com/lpt/a/2004/12/01/restful-web.html>

[13] Nelson T. H., Xanalogical Structure Needed Now More Than Ever, In: *ACM Computing Surveys, Volume 31, Issue 4,* Article 33, ACM Press, 1999. Available on: < http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html>

[14] O'Reilly T., *What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*, September 30, 2005. Available on: <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>

[15] Simmel G., *Conflict & the Web of Group Affiliations*, New York: Free Press, 1955.