Construction of Personalized Information Services for Researchers

Viktoria Foteyeva and Michail Panteleyev

St. Petersburg Electrotechnical University "LETI", Russia
{vnfoteyeva,mpanteleyev}@gmail.com

Abstract. As the amount of information available on the Internet is growing very rapidly (including Linked Data, in particular Linked Open Data), creation of customized tools for knowledge management is becoming increasingly important. The paper proposes an approach to construction of personalized information services that can be customized to the needs of different users. The approach is based on two models: model of user's information needs and information environment model (types of data sources). An implementation of the approach is considered on the example of system which is intended for concrete category of users: researchers in the area of the Semantic Web. Three basic types of users queries have been identified: news, general and analytical. Designing the personalized service is considered from the standpoint of the main stages of the queries lifecycle: its construction and execution. Two types of ontologies are used for initial query constructing: basic ontology of research activity and the domain ontology. Query execution algorithms include obtaining data from different types of sources (HTML, HTML+RDFa, RSS, SPARQL endpoints) and its processing depending on the features of the query. In addition the design pattern for effective building of queries management module is proposed. In conclusion future directions of prototype improvement are discussed.

Keywords: Personalized information services, queries to semantic data sources, information services for researchers.

1 Introduction

Currently the volumes of information available on the Internet and a variety of data representation formats preserve the tendency to rise. In this environment effective knowledge management is possible by means of personalization of information services. To do this, services should take into account individual user requests and be able to handle different types of sources, including semantic ones. It is especially important for researchers, who spend lots of time searching and processing information to stay up to date in their area of interest.

In [1] the following definition of personalized information service is given: "A personalized information service is a service towards a customer comprising (a) filtering of information out of former gathered and qualified information regarding users textual interest (b) presentation of this information using a user defined time schedule and media appropriate with recent user environment".

P. Klinov and D. Mouromtsev (Eds.): KESW 2014, CCIS 468, pp. 44–56, 2014.

2 Related Work

To help researchers to follow the news in their area (new publications, dissertations, upcoming conferences,etc.) social networks for researchers are designed, such as Academia.edu [2], Researchgate [3], Mendeley [4] etc. Since the main function is communication and search of researchers with similar interests, such systems have limited options for customizing news notifications and support a small number of predefined formats (text formats, no support for semantic sources). Search is focused on internal databases (also available in a fixed number of external ones), new sources for updates tracking cannot be added.

At present a number of approaches to semantic data aggregation are proposed. In particular, "Sigma" system [5] provides an automatic search for sources (pages with embedded RDF, RDFa, Microdata and Microformats), an integration of data from different types of sources, removing repeated data, ranking results and presenting them to user who may refine the results by adding or removing sources. ECSSE (Entity Centric Semantic Search Engine) [6] provides mashups from sources that contain structured data using large scale Semantic web indexing, logic reasoning, data aggregation heuristics and other methods. In [7,8] an aggregators of public professional events are described. The first one utilizes microblogs (e.g. Twitter) as data sources. The second one collects and integrates data in XML and utilizes RDF data model as a repository.

Some aspects of the research and development of a prototype of personalized service for researchers (including review of existing systems, the general architecture, agent-based approach to service construction) have been described in previous papers of authors [9,10,11]. Compared with other systems, the main purpose of the construction of described prototype is personalization for a particular user, the ability to extend the functionality and customization for the required information environment. This paper discusses aspects of queries preparation (on the example of news and general queries) and data collection from different types of sources, including semantic ones.

3 User Information Needs

The first step towards building a personalized service which would help to improve the efficiency of searching and processing information is to find out the information needs of users. Since different categories of users have different needs let us consider researchers as an example.

There are several human-centered models of information seeking [12], including anomalous states of knowledge model [13] in which the information needed to solve a problem are not clearly understood by a seeker. In this paper we focus on professional events to help researchers stay relevant in their area of research and plan their activities (e.g., upcoming conferences, new publications, projects, etc.), for this reason searching for answers in some specific domain is omitted. In that case users realize their information needs which in context of our prototype may be represented as a set of information queries:

$$IN = \{IQ_j\}, \tag{1}$$

where IN - information needs and IQ_j - j-th information query.

3.1 Basic Model of User Information Needs

The second step is to identify possible categories of queries and its features. The initial motivation of our project was to improve the efficiency of scientific research and educational process at the Department of Computer Engineering of SPbETU "LETI" within masters' program "Distributed Intelligent Systems". Therefore, the basic set of information needs were identified and structured based on a survey of graduate and postgraduate students and professors of the department, working in the field of Semantic web and multi-agent systems. Based on the analysis of the survey three basic categories of information queries were identified:

1. News - report about new events significant for a user. For example, an announcement of a new conference, new publication, etc;
2. General - find the set of entities with specified properties. For example, "Researchers in the field of descriptive logic", "Projects related to the Semantic Web technologies over the last 3 years", "Conferences on Artificial Intelligence in 2013";
3. Analytical - related to statistical parameters measurement ("Distribution, i.e. number of researchers, interested in the Semantic Web in the EU countries") or the dynamics ("Increase in the number of publications about the LOD for the last 5 years").

Considering selected categories user's information needs may be presented as:

$$IN = IQi = \{NQ_j\} \cup \{GQ_k\} \cup \{AQ_l\} \tag{2}$$

where NQ - news query, GQ - general query and AQ - analytical query.

News and general queries are similar, they only differ in a set of contained entities, in a set of properties (the determining factor in this case is a date) and a method of determining if the requested data is new for the user. Analytical queries do not directly contain the required data but it can be obtained by executing special operations.

3.2 Query Life Cycle

The life cycle (LC) of a user query, in general case, includes three phases:

1. query preparation (construction);
2. query execution (processing);
3. query results presentation to a user.

The life cycle phases are important for refinement of requirements to software and on the design phase.

On the query preparation phase, a user constructs a query in accordance with the selected category and sets modes of its processing. In details the preparation of a query includes:

- selection of query category;
- construction of the query;
- choosing and setting a set of external sources of data;
- setting the query processing modes (frequency and method of activation, methods of extraction / processing of required data, etc.);
- specifying form of results presentation to a user.

When constructing the query a user should specify its attributes. A news query may be generally characterized by two main attributes:

- type of a new event for user (such as the announcement of a conference, a competition, publication of a monograph, etc).
- subject (theme, topic) of event (e.g., the Semantic web, multi-agent systems).

Thus, a news query can be presented as:

$$NQ_j =< ET_j, ES_j >, \tag{3}$$

where ET_j - event type and ESj - event subject.

When preparing the general query, the user must specify the basic entity of the query (e.g. person, project, publication etc.), the topic (descriptive logic, linked data etc.), and perhaps some additional constraints (time, location, etc.). Thus, a general query can be formally presented as:

$$GQ_j =< BE_j, QT_j, AC_j >, \tag{4}$$

where BE_j - basic entity of general query, QT_j - query topic and AC_j - additional constraints.

Furthermore, on the step of query construction user forms a list of sources to be processed at each cycle of query implementation.

On the query execution step it is processed in accordance with the specified execution mode. Query processing generally includes:

- collection of information from sources;
- selection of required facts from sources;
- further processing according to the query type.

On the last step the query result is presented to a user.

3.3 Model of the Information Environment for the Personalized Service

To provide flexible customization of the service to the dynamically updated global information environment it is necessary to determine the model of the environment.

This model should describe the properties of the environment, such as types of sources, protocols to access them, formats of queried data, etc. The Internet contains a huge number of distributed heterogeneous sources and the way they may be collected depends on their type. In our project the following types of sources are selected for consideration:

1. News RSS feeds.
2. SPARQL-endpoints.
3. Pure HTML web pages (without microformats, microdata or RDFa).
4. HTML + RDFa (or with microformats or microdata).

Thus, basic data for general or news query are:

- a query;
- a list of external sources.

A number of popular sources of scientific information, in particular [14,15,16], provide the opportunity to proactively inform users in accordance with specified requirements. In our project we are planning in particular provide a single interface to configure them.

4 Architecture of the Personalized Service and the Used Ontologies

As shown on Fig. 1 the main modules of the prototype are the web application, the client and service agents and a knowledge base (containing three types of ontologies).The prototype that is developing as a Java web application is at an early stage. Ontologies are stored in PostgreSQL database. To deal with ontologies Jena framework [17] is used. Agents are built with the help of Jade framework [18].

Since the service prototype for researchers aims to help them in their everyday work, types of ontologies were selected based on the analysis of their activities.

Ontology of informational environment (IE). Describes the information environment in which the service works: data sources, document formats and access protocols to them (based on the model of information environment). Due to ontology of informational environment and the service architecture user have the ability to choose and add new types of sources.

Basic ontology of research activities (BRA). Contains information about interesting to user events and describes infrastructure of scientific and educational activities (based on the categories of information queries for this group of users). This ontology allows users to customize the types of followed up events and related data (e.g., for the event “publication of a new paper” data about authors are related).

Domain ontology (DO). Describes the structure of a particular domain and provides a flexible configuration of the service. Due to the domain ontology, service is not tied to a particular domain and may be customized to the user

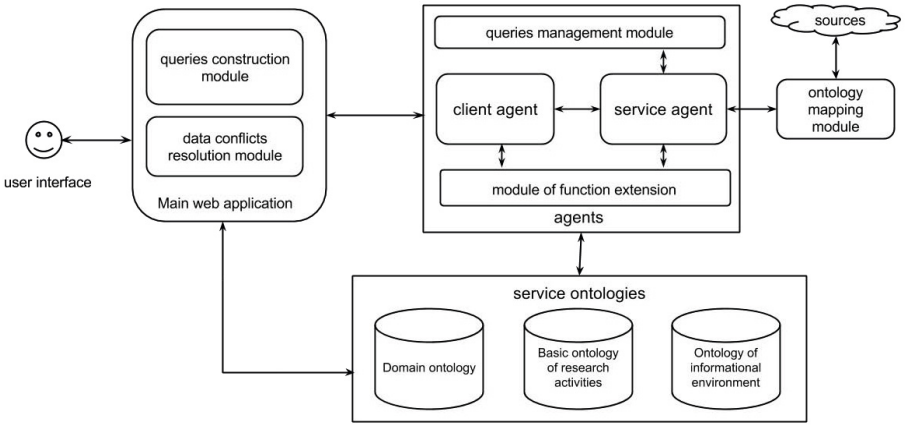

Fig. 1. General architecture of the prototype

interests. To do this during the setup phase of the service ontology mapping is required that is quite a tedious task, but it may be partially automated.

For each user customized versions of ontologies of three types are created (user ontologies). Query is constructed using described ontologies: types of events are selected from the BRA, query themes from the DO, data sources from the IE. Service agent is an important module of the prototype which is used to construct queries and collect data from different types of sources.

5 Query Management Module

For efficient processing of user queries a module should be constructed which main objectives would be:

- to make the process of getting query result transparent for service agent regardless of data source type;
- to provide an ability to add new types of sources without changing the architecture of the service.

This module (called "Query Management Module") is used in several ways:

1. On a query processing step. The input of the module is a query q of appropriate type (in the prototype - SPARQL or a string query) and a type of source or a particular source s (if specified). As a result, the module returns triples T containing the result data. The module defines the access method to the source of a particular type: it sends a query to the appropriate source, receives the response and brings it to a format suitable for storage in the user ontologies (triples). The module has two work modes: user queries processing in real time and scheduled data updates collecting (e.g., once a day).
2. On a query construction step to the source of a specified type - for configuration and the query's initial data retrieve.

At the architecture level this module is located in the service agent.

Next, the first two steps of queries life cycle are considered: its construction and data collection from the selected types of sources, and implementation of these steps using the queries management module.

6 Query Preparation

6.1 SPARQL Query Preparation

For scientific and educational purposes it is proposed to select a list of typical SPARQL queries with the possibility of user configuration for a specific endpoint. Typical queries are constructed in terms of the service ontologies. By configuration we mean the imposition of restrictions on selected by user properties available in a specific endpoint.

The algorithm of query preparation:

1. User selects a typical SPARQL query and an endpoint. Ontologies used in the endpoint and the service ontologies should be previously mapped. In this case it is possible to select an entity class with meaning similar to the ones described in a typical query using such properties as sameAs, skos:closeMatch and skos:exactMatch for a particular endpoint. For this task ontology mapping module is used.
2. Queries management module selects all the properties for the requested entity class from the selected endpoint. For each query a binding to a basic entity is stored (BE for general query or ET for news query, see 3.2).
3. User selects the properties on which he would like to put restrictions.
4. Types of selected by user properties (e.g., string, date) are defined.
5. User is asked to input values of selected properties as restrictions for the query.
6. Ready to run query is stored in the knowledge base of the service.

Let us notice that ontology mapping is going to be done in semi-automatic way by the user since nowadays the process cannot be fully automated. Currently COMA 3.0 [19] is used for mapping in the prototype. Various algorithms may be used for this purpose and every year Ontology Alignment Evaluation Initiative presents comparison results of the best ones (e.g., results of 2013 year [20]).

6.2 News Feeds (RSS)

To get updates from the RSS-sources RSS-aggregators are commonly used. Data collection from this type of source has little scientific interest, but as the format is very common its support is added to the service prototype. So far RSS has two most popular versions: 1.0 is based on XML standards and RDF, and the 2.0 has a simpler syntax and is not an RDF-format. Version 2.0 may be converted to the RDF using XSLT.

To provide the ability to process SPARQL queries for this type of sources an initial extraction of the RSS-feed should be performed on the step of adding the source to the service for:

– mapping between dictionaries (RSS modules) and services ontologies;
– storing data about feed content in the service (used classes and properties).

The initial extraction is performed by the queries management module. After that the algorithm of a query creation to the news feeds is similar to the one described in Section 6.1.

6.3 HTML+RDFa

In order to process SPARQL queries to HTML+RDFa documents an initial extraction of the RDF-triples from the document should be performed. To do this various libraries for RDFa extraction may be used in the query management module (in the prototype Semargl [21] was used). After triples extraction the information about ontologies used for defining the RDFa will be saved. Next task is to build a SPARQL query. To do this it is necessary to map the entities of the typical query and the entities used in RDFa. The algorithm is similar to one described in Section 6.1.

6.4 HTML Documents

For this source type there are two basic approaches:

– processing using a structure of a particular site, i.e. the specific predefined HTML markup of the site.
– general processing without considering the peculiarities of a particular site. In this case the text queries are applicable. Selection of news facts may be performed using text mining.

In this paper only the first case is considered. In our opinion the optimal way would be to offer users to add their own markup to pages (e.g., RDFa or microformats). For convenience, an interface to help a user add a markup to elements of HTML pages containing specific information (such as conference title, start date, etc.) is required. This approach brings HTML processing to HTML+RDFa case. The most well-known tool that works in a similar way is Structured Data Markup Helper [22] from Google (supports schema.org, partially JSON-LD and microformats).

In the service prototype the following steps should be completed for querying HTML-pages using the proposed method:

1. At the step of adding HTML page user marks it by adding RDFa properties using the service interface.
2. Markup for this page is stored in the service.
3. After that user creates queries the same way as he does it for HTML+RDFa.

7 Data Collection Using the Query Management Module

Data collection from sources of different types has its own features considered below.

7.1 Data Collection from SPARQL Endpoints

The module of queries management in order to work with sources of this type should provide execution of SPARQL queries based on ETj and ESj, taking into account the features of the specific endpoint. Processing algorithm of this type of source is the following:

1. execute the query for a given SPARQL-endpoint;
2. get the triple from the SPARQL-endpoint;
3. save results to user ontologies.

To save the triples to the user ontologies a particular query pattern should be used which universally chooses a subject, object, predicate and a text label if available. The pattern of the general SPARQL-query for data collection is the following:

```
SELECT ?subj ?prop ?obj ?label
WHERE {
   ?subj a prefix:OntologyClass.  #event type
   #get property to filter value
[  ?subj prefix:OntologyProperty ?property. ]
   ?subj ?prop ?obj.
[  OPTIONAL  { ?objrdfs:label ?label.}
   #filter property value
   FILTER regex(?property, "propertyValue", "i")   ]
}
LIMIT N
```

Optional parts of the query are in square brackets. Values filtering is held in different ways depending on the type of values in the query (in the example by string value).

7.2 Data Collection from RSS Feeds, HTML+RDFa Documents and HTML

Since for a particular RSS-feed and a HTML+RDFa document SPARQL queries were constructed during the setup step, data collection from these types of sources is described in 7.1.

In the service prototype the following steps should be performed for the collection of data from HTML documents:

1. the queries management module downloads a page and adds the saved RDFa markup (see 6.4);
2. the queries management module collects data from HTML+RDFa (with the help of external libraries integrated into the prototype) using a predefined user queries.

8 Design Patterns for Data Collection

To be effective the implementation of the queries management module should provide (see Section 4):

- abilities for adding a new type of data source without changing existing classes in the module;
- separation of service agent class from the specific implementation of source classes: service agent should receive and forward a query but should not have the information about how to get data from the source of a particular type.

Analysis of software design patterns showed that it is advisable to adapt the pattern Command [23] for data collection in real time (Fig. 2). For each operation (e.g., retrieve data from a source by a query execution, get all properties of an entity class) of a particular source type (SPARQL endpoint, RSS, HTML+RDFa, HTML, search engine and semantic search engine) classes are created (e.g., the class for retrieving data from a SPARQL endpoint is "GetFromEndpointCommand"). They contain instances of the corresponding source class (let us call it a "controller" of the source, Fig. 2 shows "SPARQLController" as an example). The "SPARQLController" class contains all the methods for retrieving data from a specific source type (a SPARQL-endpoint). A "RequestDistributor" class is responsible for the distribution of user requests to appropriate controllers.

For scheduled data collection it is important to set a sequence of queries for update collection and maintain high performance. The command pattern can be used to handle a situation where there are a number of jobs (commands) to be executed but only limited resources available to do the computations [23]. In this case objects that implement the ommand interface are queued and program threads sequentially extract commands and call their execute() method, and after that go back for the next command object.

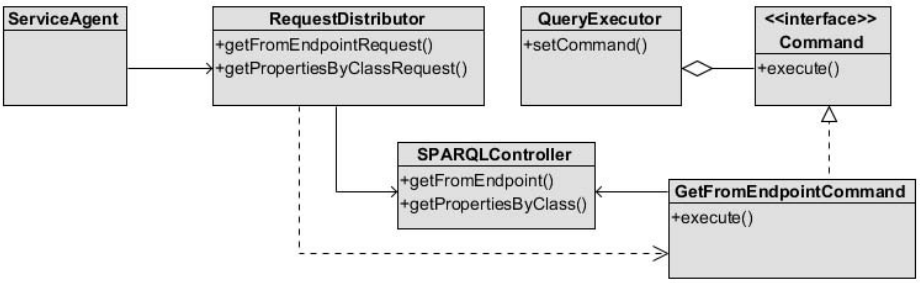

Fig. 2. Class Diagram of "Command" pattern for data collection

9 Preliminary Evaluation

The prototype was tested by a small group (7 people) of postgraduates and students enrolled in the master's program "Distributed Intelligent Systems".

Under this program the disciplines “Semantic Web” and “Multi-Agent Systems” are taught. Students would like to know more about these areas, e.g. about publications, conferences, ongoing projects, organizations and people who work in these areas, and get some data analysis (e.g., how many papers were published over the last few years on some particular topics, what are the dates to apply for a workshop, etc).

To do this, they should specify a list of data sources that can be set manually or selected from the list proposed by a service. After that the queries to structured data sources should be formed. As a result, on the main page of the application user sees aggregated data from predefined sources. The user may choose how to visualize the data: by informational blocks (publications, conferences, organizations, people, projects) with summaries (e.g. the number of found conferences) or an adapted RDF-graph. The user may choose the informational

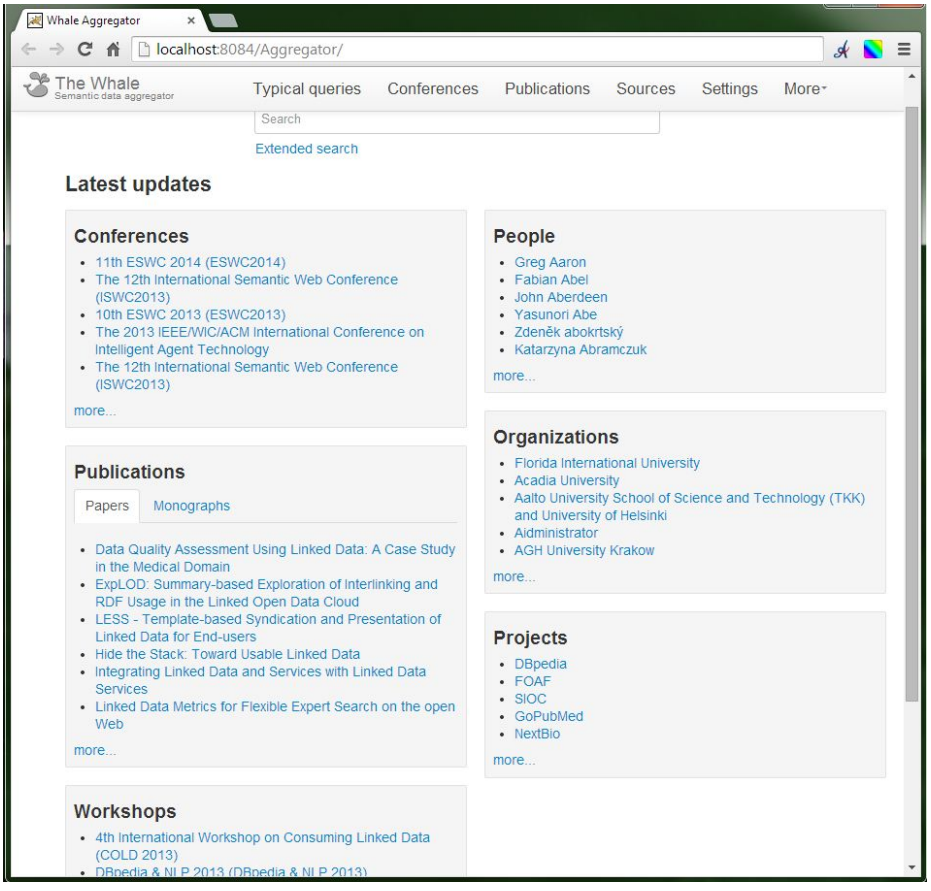

Fig. 3. Main page of the prototype which represents a few latest results of each category (conferences, publications, workshops, people, organizations and projects)

block he wants to know more about and get more detailed information on the page of the prototype dedicated to it. After aggregating data for the first time the user may set properties for updates aggregation to stay current.

As part of the test scenario the prototype was preconfigured to aggregate news about intelligent agents and the Semantic Web. Main page of the prototype, which represents a few latest results of each basic entity (conferences, publications, workshops, people, organizations and projects) collected using all predefined queries, is shown in Fig. 3. At the end of evaluation period (two weeks) the students confirmed that:

- the prototype saves time: all updates from sites without RSS-feeds were available without the need to visit them one by one;
- it is convenient to access RSS-feeds from the same interface;
- they were offered some publications on the predefined topics from SPARQL-endpoints which they had not previously considered as in-formation sources because of usage difficulties.

10 Conclusion and Future Work

To create a personalized service we have structured the information needs of specific categories of users and the information environment consisting of different types of sources. The prototype provides the ability to collect information in a variety of formats for the needs of a particular user and due to the ontological approach may be customized to the required information environment. However for now there are some difficulties in customization to the needs of a user. For that reason further development is needed, in particular in construction an interface for SPAQRL queries, more high level of automation of ontologies mapping, client agent configuration, etc).

In future we are planning to implement a conflict resolution module, an interface for SPAQRL queries construction for non-expert users and to add the ability to handle HTML microdata and microformats. The personalized service should also be tested over a wide range of users and its effectiveness should be evaluated with more rigorous metrics.

References

1. Ritz, T.: Personalized information services: An electronic information commodity and its production. In: Proceedings of the ICCC/IFIP Conference, pp. 48–59. IOS Press, Amsterdam (2001)
2. Academia.edu official site, `https://www.academia.edu`
3. ResearchGate official site, `http://www.researchgate.net`
4. Mendeley official site, `http://www.mendeley.com`
5. Sigma official site, `http://sig.ma`
6. Cyganiak, R., Catasta, M., Tummarello, G.: Towards ECSSE: live Web of Data search and integration. In: Semantic Search 2009 Workshop, Madrid (2009)

7. De Vocht, L., Selver, S., Ebner, M., Mhlburger, H.: Semantically driven Social Data Aggregation Interfaces for Research 2.0. In: 11th International Conference on Knowledge Management and Knowledge Technologies, pp. 43:1–43:10. ACM, New York (2011)
8. Al-Safadi, L., Alkhatib, N., Babaier, R., Assum, L.: Semantic Aggregator of Public Professional Events. J. of Applied Sciences 12(7), 653–660 (2012)
9. Panteleyev, M., Foteyeva, V.: Building aggregator of scientific and educational data for Semantic Web. In: 3rd Conference on Knowledge Engineering and Semantic Web, St. Petersburg, pp. 73–79 (2012)
10. Foteyeva, V., Panteleyev, M.: Agent-based semantic data aggregator. In: 4th Conference on Knowledge Engineering and Semantic Web, Book of abstracts, St.Petersburg, pp. 22–24 (2013)
11. Foteyeva, V.: Problems of building semantic aggregators. In: Proceeding of International Conference on soft Computing and Measurements, St.Petersburg, pp. 110–112 (2012)
12. Marchionini, G.: Information seeking in electronic environments. Cambridge University Press, Cambridge (1997)
13. Belkin, N.J., Oddy, R.N., Brooks, H.M.: ASK for information retrieval: Part I. Background and theory. J. of Documentation 38(2), 61–71 (1982)
14. Scopus official site, `http://www.scopus.com`
15. Google scholar official site, `http://scholar.google.ru`
16. WikiCFP official site, `http://wikicfp.com/cfp`
17. Jena framework official site, `http://jena.apache.org`
18. Jade framework official site, `http://jade.tilab.com`
19. Coma 3.0 - a schema matching system official site, `http://dbs.uni-leipzig.de/Research/coma.html`
20. Grau, B.C., Dragisic, Z., Ecker, K.: Results of the Ontology Alignment Evaluation Initiative. In: 12th International Semantic Web Conference, pp. 61–100 (2013)
21. Semargl official site, `http://semarglproject.org`
22. Structured Data Markup Helper, `https://www.google.com/webmasters/markup-helper/`
23. Freeman, E., Robson, E., Bates, B., Sierra, K.: Head First Design Patterns. O'Reilly Media (2004)