MultiBeeBrowse Accessible Browsing on Unstructured Metadata

MultiBeeBrowse – Accessible Browsing on Unstructured Metadata *

Sebastian Ryszard Kruk[1], Adam Gzella[1], Filip Czaja[1,2], Władysław Bultrowicz[1,2], and Ewelina Kruk[1]

[1] Digital Enterprise Research Institute, NUI Galway, Ireland
[2] WETI, Gdansk University of Technology, Poland
<firstname.lastname>@deri.org

Abstract. Growing abundance of information on the Internet, especially the Next Generation Internet, poses even more challenges on more efficient information management; hence it has brought attention of the researchers to the faceted navigation. Existing solutions, however, do not address the majority of users, who are still inexperienced in using the faceted navigation solutions or who do not understand underlying concepts of the Semantic Web technologies, or both. The query refinement process, while using the faceted navigation interface, is more complex than, e.g., refining a simple keyword-based query.
In this article we present MultiBeeBrowse (MBB), an accessible faceted navigation solution that solves aforementioned problems in the browsing environment. We present how to improve users' access to their history of refinements; we discuss how users can share their browsing experience. And last bust not least, we present an adaptable user interface, which aims to decrease information overload.

1 Introduction

Future Internet, as envisioned by both the academia and the industry, will be the Web of metadata. This large, metadata rich information space, will be hardly manageable with current techniques. Until machines will really understand what we mean, we will have to refine our queries ourselves. The Semantic Web community investigates new solutions for the faceted navigation designed for the unstructured metadata; in contrary to, e.g., Flamenco [18], new interfaces are designed to facilitate browsing without prior knowledge of the structure of the information space.

* This material is based upon works supported by Enterprise Ireland under Grant No. ILP/05/203 and by Science Foundation Ireland Grant No. SFI/02/CE1/I131, and partially by KBN, Poland under Grant No. 4T11C00525. The author would like to acknowledge Daniel Schwabe, Peter Brusilovsky, Bill McDaniel, Henryk Krawczyk, Stefan Decker, the DERI eLearning Cluster, the Corrib.org working group for fruitful discussions, and the group of 20 patient volunteers who agreed to take help with the evaluation.

1.1 Motivations

In the faceted navigation the information space is partitioned using orthogonal conceptual dimensions of the data. These dimensions are called facets and represent important characteristics of the information elements. Each facet has multiple restriction values and the user selects a restriction value to constrain relevant items in the information space. The faceted navigation interface allows to quickly narrow down number of results by choosing more and more precise values from various angles. The facet theory [13] can be directly mapped to navigation in semi-structured RDF data.

The faceted navigation aims to allow users to harness the full potential of the web rich in semantics. Many people, however, have still problems with using keyword search. In 2004, OneStat.com[3] announced that 77% of queries consists of 3, or less, word phrases. In January 2007, RankStat.com[4] reported that 80% of queries consist of 4, or less, word phrases. It means that over last 3 years only a few people started using more complex queries. A query consisting of two word phrases is the most popular, which means that most users do not create complex queries; they rather rely on their luck, reading through long lists of results, and eventually on refining they queries.

Constructing more complex (5 and more words) keyword-based queries still poses problems to an average user; therefore, we cannot expect that advanced navigation, e.g., faceted browsing, as it is delivered at the moment, will gain much attention in the near future.

There are also two other features which still make it easier to construct keyword queries composed to faceted navigation: (1) refining a query by adding, removing, or changing certain words is relatively easy, since we operate in one dimension of vocabulary definitions; we need to remember not only values (keywords) but also the names of the facets (properties). This can become a real nuisance when we trying to refine a query many times, and when by going back in the refining process we loose our previous refinements. (2) we can easily share our keyword-based searching experience with others, by dictating the words, giving a hint on them, or simply copy-pasting the URL from a search engine. The same operation using the faceted navigation is much harder to achieve; it takes much longer, e.g., to dictate all operations a user is required to perform.

In this article we present an accessible browsing solution, a MultiBeeBrowse component for collaborative faceted navigation, which answers aforementioned motivation requirements. By *accessible browsing* we understand a user interaction with the browsing services, where accessibility is achieved through: improving access to the history of refinements, adapting the presentation and browsing style, lowering the information overload, and engaging users in the collaborative browsing. The term *unstructured metadata*, used in this article, adheres to the concept of information, which structure, or schema, has not been defined before, or could not be identified while processing the information.

[3] http://www.onestat.com/html/aboutus_pressbox27.html

[4] http://www.rankstat.com/html/en/seo-news1-most-people-use-2-word-phrases-in-search-engines.html

1.2 Related projects

With the growth of the Semantic Web we need more powerful tools to search and browse on the unstructured metadata. We have identified four types of the navigation interfaces: (1) with a keyword search, e.g., Swoogle [5], (2) with an explicit querying, e.g., Sesame[6], (3) with a graph visualization, e.g., IsaViz [7], and (4) with faceted navigation, e.g., Longwell[8].

Oren *et al* [12] showed the advantages, such as speed of information retrieval and the intuitiveness, of the faceted navigation. They presented and evaluated, both formally and experimentally, many solutions in this area such as BrowseRDF [12], Flamenco [18], mSpace [14], Ontogator [6], Spectacle[9], and Siderean Seamark[10]. BrowseRDF turned out to be the most complete and powerful in the whole group. Nevertheless the authors omitted one of the best, in out opinion, faceted navigation solutions: SIMILE's Longwell. Another omitted, but very interesting system, is /facet [5] browser which was created and presented at the same time as BrowseRDF.

Longwell is a navigation solution, which builds upon flexibility of the RDF data model and effectiveness of the faceted browsing user interface paradigm. It allows to visualize and browse arbitrary RDF dataset; users can quickly build easy to use web sites, based on their own data.

BrowseRDF is minimalistic in form; Longwell, have a strong arsenal of functions, and at the same time stays user-friendly, looks good and is reasonably fast. It is also quite similar to /facet solution. /facet has most of the Longwell features and a special mechanism to deal with the large number of facets. However, Longwell is more flexible, fully configurable, when displaying the results.

One of research areas, related to the topic presented in this article, focuses on graphical representation of the queries on RDF graphs. Harth *et al.* [4] presented an interesting solution, which could be used to augment existing services, such as BrowseRDF, Longwell, /facet, or MultiBeeBrowse, with SVG-based graphical representation of faceted browsing queries.

Since we lacked an explicit comparison between Longwell and BrowseRDF we decided to evaluate these two solutions together with our own (see Sec. 7). To keep the evaluation easy and clear for users taking part in it we decided not to evaluate /facet as this solution is quite similar to Longwell.

1.3 Outline of the paper

In the next section we present an overview of browsing operations on interconnected metadata. In section 3 we present how browsing context information can be delivered through a zoomable interface paradigm. Section 4 shows how

[5] http://swoogle.umbc.edu/
[6] http://www.openrdf.org/
[7] http://www.w3.org/2001/11/IsaViz/
[8] http://simile.mit.edu/wiki/Longwell
[9] http://www.aduna-software.com/products/spectacle/
[10] http://www.siderean.com/

search and browsing experience can be exchanged among the users. In section 5 we identify adaptive hypermedia, and other techniques, for improving accessibility of browsing and of presented results. Finally, in section 6 we identify services oriented architecture of our solution, MultiBeeBrowse. Eventually, we report results of our user-based evaluation in section 7.

2 Browsing on Interconnected Metadata

Information discovery is a key capability of modern, information society; growing abundance of information accessible through various media, including the Internet, has always been both the blessing and the curse of the contemporary humankind. When Google showed up, it simple swept away other search engines with its PageRank algorithm. But the new era of the Internet, which grows to become a network of social media and interconnected metadata (semantics), brings new challenges in the field of the information retrieval.

This section presents an overview of different information discovery techniques, especially in the context of semantic and social web.

2.1 From Searching to Filtering

People, at the moment, are getting the grip on the keyword-based search very slowly (see Sec. 1.1); therefore, more complex navigation solutions are still beyond capabilities of the majority of the Internet users. When dealing with metadata information a more advanced search can come handy. In a digital library we can specify that *Kruk* should be matched against names of the authors only, and *navigation* should be found in titles of articles only. A similar feature is offered by Google search; users can restrict their search to a given site only, using a *site:* prefix of the URL of the site. Although more powerful, these advanced search solutions are not very popular and users tend to stick to simple keyword-based queries. Nielsen [11] even suggests to discourage novice users to using advanced search features. Another way to extend search operation is by delivering a natural language query interface. Users can ask complex queries by using natural language sentences [9].

A simple filtering user interface is a step towards better navigation using advanced search capabilities; a user can select a value for each property (facet) from a list of possible values for this property in the given context. This approach seemed to gain more appreciation from the users [15]. It gave incentives to further research in the area of faceted navigation; the Flamenco project [18] is one of most famous results.

2.2 From Filtering to Browsing

The new Web is not just a network of resources and shallow metadata; it is a network of highly interconnected metadata, often called semantics. The new Internet is also a social medium, with contribution from all users. In the Semantic

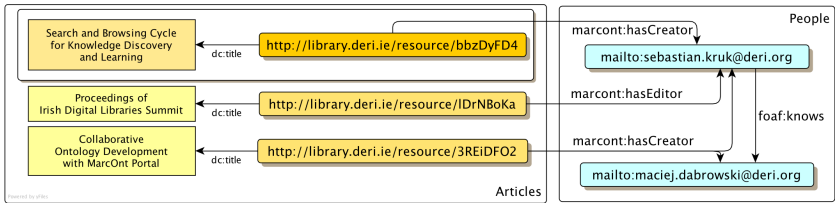


Fig. 1. Example of part of an RDF graph with article and person concepts

Web, this interconnected information is represented as an RDF Graph [10]. This imposes new challenges on the navigation techniques. It is required not only to filter but also to browse from one set of results (of one type) to a new set of results (of different type).

In our example (see Fig. 1) we have two types of resources: articles and people. A typical scenario for browsing on this information set would be to find other articles written by authors of, e.g., *Search and Browsing Cycle for Knowledge Discovery and Learning*. In our case this could be realized by browsing from this article, to a set of its authors, and browse back to the list of their publications. If we analyze the example graph (see Fig. 1) we will notice that *Proceedings of Irish Digital Library Summit* will not be returned in our browsing process, since the resource is related to the person with *hasEditor* property. In this case, we would need to be able to find similar resources matching values (in this case a person), not properties (*hasCreator*).

This simple scenario illustrates 3 requirements for a browsing on interconnected information:

1. ability to browse from one set of results to another set of results, of possibly even different type, related through a given property;
2. ability to represent, in human readable form, inverse properties (see Def. 1) to those in initial graph of information;
3. ability to find similar resources based on either properties or values related to given set of results.

The first requirement is the basic features of each navigation solution designed to operate on arbitrary information. It allows to find resources related to these in current set. Hence, we can get a list of co-authoring people, based on the initial set of some of their publications.

Definition 1 (Inverse property). *In RDF Graph G = (V, E, L, l), for each* $e \in E$, $v_1, v_2 \in V$, *label* $\in L$, *so that e:* (v_1, v_2) *and* $l(e) = label$, *an Inverse Property is a defined as* $(e_{inv}: v_2, v_1)$[11], *and* $l(e_{inv}) = inv(label)$, *where inv:* $L \rightarrow L'$ *is an arbitrary labeling function*[12]

[11] In most cases the inverse property is a logical concept, which is not explicitly provided in the source RDF graph, but can be explicitly defined using, e.g., `owl:inverseOf`

[12] Inverse labeling usually involves natural language processing

Usability of the aforementioned browsing feature relies on realizing the second requirement - being able to traverse the graph backwards to the direction of given property. Support for inverse properties (see Def. 1) relies on both modeling of possible interactions with given set of information, and on delivering user-friendly names for the inverse properties. In our example (see Fig. 1), a user who wants to go from a set of articles to a set of authors, will look for *has creator* property; while, a user who wants to go from a set of people to a set of article, authored by these people, will look for *is creator of* property. In most cases delivering a separate label for the inverse property (in our case *is creator of*) improves the overall usability of the browsing solution (see Sec. 5). The navigation interface should allow for a seamless representation of underlying graph representation of data, without requiring users to understand the technicalities.

The third required browsing operation is finding similar resources based on either given property or on given value. In the first case, it would mean to look for other articles co-authored by the same people who wrote any of the articles from the first set. In the latter case, this would mean to look for any resource somehow[13] related to given value. It would allow for finding publications either authored or edited by given person, but would not restrict the final set of results to the same type of resources as the initial one.

When dealing with resources that can have different properties, such as *has creator* or *has editor*, users might also want to restrict the set of presented results to all resource that have given property, or to those that do not have given property at all.

Some of more complex queries requires couple of steps to build; when you what to say that you are looking for articles that were *presented at a conference* which *took place in Ireland*, and whose *authors* were *members of DERI*, you need to compute queries: (based on conference location and affiliation of authors), and join them together. We have identified 4 types of combination operations:

- **intersection** – each resource must exist in both input sets;
- **sum** – each resource must exist in either of input sets;
- **difference** – each resource must exist in the first set, but not in the second one;
- **binding** – returns a set of results that span together resources from both sets, with a chain of connections not longer than a given parameter.

Access to resource metadata In most cases, search and browsing services would rather present the result itself; in the context of multipurpose navigation framework, however, access to the resource metadata allows users to build queries on this seed information. They can find similar resources (see Sec. 2.2), or continue with filtering and browsing.

[13] New results need to have at least one property with a value matching the given one

3 Zoomable Browsing Context

Creating a faceted browsing query requires a number of refinement steps (see Lemma 1); each of these steps involves one of atomic operations (see Def. 2) defined in previous section (see Sec. 2).

Definition 2 (Browsing Operation). *A Browsing Operation $bo \in BO$, is the simplest operation which can be invoked on the navigation engine; it takes output of zero, one, or two other Browsing Operations as input*[14]*; $BO = \{Resource_0, Search_1, Filter_1, Browse_1, Similar_1, Combine_2\}$, each Browsing Operation represents one refinement step in the browsing query building process.*

Lemma 1 (Decomposition of Browsing Query). *Each search and browsing query Q can be decomposed into a set of consecutive browsing operations, each representing one refinement step in the query building process.*

Mastering the right query can take some time, especially to novice users; sadly current web browsers does not handle history of actions as a graph of consecutive operations, but rather as a flat list of *recently* opened URLs. Therefore our exact refinement process is lost. When we decided to use backwards button to go back couple of refinements steps, and continue with refinements, our previous history of refinements is only accessibly through a meaningless view of all previous URLs visited with our browser. What if we would like to retrieve our browsing actions from long time ago, and be able to *replay* these refinements, step by step?

Based on these two goals: handling many paths of back and forth refinements, and access to a structured history of operations, we have identified 4 views of browsing context, which allow user to access effortlessly each of aforementioned features.

- *Basic browsing view* provides access to all browsing operations (except for *Combine*) with a typical search and browsing user interface (see Fig. 3). To further enhance usability of our solution, we have extended the query building part with suggestions of properties and values, and results rendering part with an in-situ browsing menu (see Sec. 5).
- *Structured history view* allows users to view their current results in the context of previous and following (if any) operations. This view is almost the same as the *basic browsing view*, with one difference that users see 6 slots (see the *Honeycomb^{TM}view*) with previous operations, and another 6 slots for further browsing. Some of the slots might be already occupied with definition of previously performed refinements; other are free, and a browsing pop-up allows to specify which slot to use to continue browsing (see Fig. 3).
- *Honeycomb^{TM} view* presents users a comprehensive overview of their current browsing context. Each browsing query is represented with a hexagon lozenge in a 3D visualization (see Fig. 3); some edges between hexagons represent

[14] Numbers in subscript, next to the name of operation, indicate a number of other browsing operations which can be prepended to given one

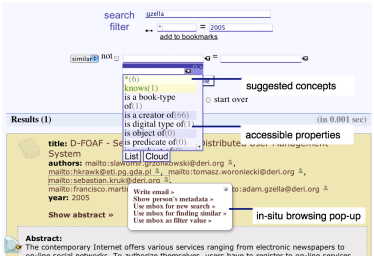


Fig. 2. Basic browsing view

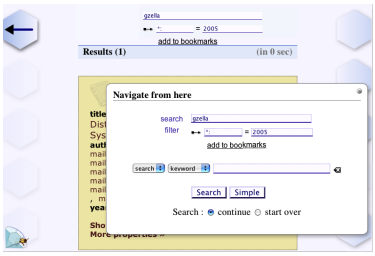


Fig. 3. Structured history view

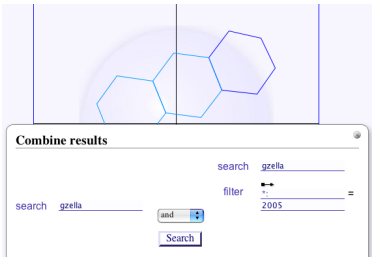


Fig. 4. Honeycomb^{TM} view

Fig. 5. Life-long history view

browsing operations that were added to the chain of operation to create a new browsing query. With this view, users can get a quick overview on their browsing session; they can quickly jump to a selected browsing query in the current context. This view also allows to perform *Combine* operations, by selecting one of vertexes with exactly two adjacent hexagons. In a similar way, by selecting an edge with only one adjacent hexagon the user can perform any browsing operation, which was allowed in previous views.

– *Life-long history view* presents all previous sessions in which the given query was invoked (see Fig. 3). It allows a user to quickly *move in time* to some browsing context, review refinements invoked after the given browsing query. User can even jump back to that context and continue browsing.

These four views have been set up, so that users can *zoom out* from the view of results, to a view of the browsing context, to a view of all browsing contexts (life-long history).

4 Collaborative Browsing

In the standard keyword-based search queries are not complicated. They usually consist of three or four words (see Sec. 1.1), so it is very easy to recall and repeat searching process. Sharing such queries is also quite easy as it is enough to dictate or send the set of keyword or a URL that invokes the search query. With faceted navigation querying process is not always straight forward. It often consist of keywords and some actions that depends on the previous ones and the

context. Thus, sharing queries and results of them is much harder with faceted navigation approach.

Definition 3 (Collaborative browsing). *In group of users U and browsing queries sets B_m, for u_x, $u_y \in U$ and set of browsing queries b_{1_m}, b_{2_m}, ..., $b_{n_m} \in B_{u_m}$, where m ∈ (x, y). Collaborative browsing is a process in which u_x browse through the B_{u_y} and uses $b_i \in B_{u_y}$ Optionally u_x refines query $b_i \rightarrow b_i^{'}$ and performs $B_{u_x} = B_{u_x} \cap b_i$ and/or $B_{u_x} = B_{u_x} \cap b_i^{'}$*

Collaborative browsing (see Def. 3) solution, presented in this paper closes the aforementioned gap between classic keyword-based and faceted browse. All the actions in the system, facets dependency and the context of the search are saved in a special URL (see Sec. 6). By invoking such a URL users can retrace all their browsing steps and get desired results. It is also possible to refine the query if the result is insufficient or the search objective has changed.

Users can now share these specially constructed URLs to the faceted browsing results just like they did with keyword-based search. They can send them by e-mail, through instant messaging, or put it on the web site. But searching for URL in e-mails, browsing instant messaging history, looking for web page is not very convenient and creates lots of problems; user would rather perform new browsing than use the existing one, no matter how good it was. The navigation component can be extended with an online bookmarking solution, such as SSCF (Social Semantic Collaborative Filtering) [7]. SSCF allows users to create the knowledge repository by bookmarking and categorising interesting and valuable resources. Users can utilise the knowledge and expertise of others by browsing and importing resource gathered by their friends. Resources in SSCF are semantically annotated with community established categorization. Users are also able to set fine-grained access rights to each category. SSCF has been extended to deal with URLs representing browsing queries. They became standard SSCF resources, so users can now share browsing experience with ease and without any hassle.

To make the collaborative browsing truly happen, while browsing resources users have quick access to their SSCF bookmarks. They can specify a name and a description and select where in the hierarchy the current browse query be added. It becomes an SSCF bookmark and can be shared by the community. The browsing query has the same features as a standard SSCF resource; it can be copied, cloned, imported by a friend or put in a directory with restricted access rights.

Users can share their browsing queries, and by bookmarking refinements of these queries, they actually perform collaborative browsing. With collaborative browsing people can share their knowledge and their expertise with the social network. Users can also help each other and learn how (by exchanging query bookmarks) to get the satisfying results from the search process.

5 Adaptable Browsing Interface

In this section we will present how adaptive hypermedia, and other personalization techniques, can improve the accessibility measures, such as the speed and the ease of use, of navigation process. An accessible navigation interface contains elements that are automatically adapted by the system according to user's profile, or that can be dynamically changed by users through their actions.

5.1 Results presentation

As a result of executing the given query, accessible browsing interface returns a set of resources together with all available metadata and RDF relations. A predefined rendering style can be applied based on the value of `rdf:type` property. Each type has some main properties; these properties group predicates describing the same concepts within different ontologies. We have also identified descriptive properties of the most popular resource types, e.g., an *abstract* for an *article*. We use the adaptive hypermedia strechtext technique [1] to present on demand additional, collapsed by default, information, including descriptive properties. A generic result is rendered with only a label, with all other properties collapsed. The label for a generic result, as for all properties and other resources, is generated based on values of `rdfs:label`, `dc:title`, or `foaf:name` properties. Also the URIs of namespaces are abbreviated to short names. This feature adds an aspect of adaptability to the result presentation interface.

5.2 Using results for refining queries

Most of the values displayed in the results page, e.g., property names and values or the resource URIs, can be used for refining user's query. Instead of typing some literals or URIs in search box users can click on the given value and use it for building their query using *in-situ browsing pop-up*; they can start new search or simply load the chosen URI. The action choice box that appears after clicking a value offers the same functionality as the main search box. It limits, however, possible options to only these actions, where the clicked value can be used. It means that users can not load a resource using a literal value or they can not choose a value that is not a predicate for further browsing. System recognizes what element was chosen (property, value, or resource) and adapts the action box by changing the set of possible actions for a given element. This technique is one of the ways of limiting information overload in the browsing solution.

5.3 Adaptive concepts suggestion

Whenever users choose an action type and wants to specify some values for this action, such as keywords or predicate name, the list of possible values appears, and they can use one of the suggested entries. A given number indicates the amount of results returned when using this value is presented next to each entry.

The list of the results can be presented as a regular list of entries or as a tag-cloud, depending on user preferences.

One of the adaptive aspects of our browsing user interface is suggesting to the user the values while building queries. When browsing through the results or building new queries system suggests the values that user is most likely to use.

The algorithm for suggesting the entries is based on the statistics concerning most popular users choices, most relevant predicates in given context, and the history of browsing saved for a given user. User is not limited to the suggested predicate values; they are only a hint to make the queries building easier. The suggestions appear on the top of the list of possible entries, in different color than the rest of entries. Up to three most likely to be used values are presented.

5.4 Accessible predicates names

The faceted browsing user interface is meant to be used by generic users that do not necessary have to be familiar with concepts of RDF. In order to make building of search or browse queries easier we decided to create a kind of "accessible predicates names". At the moment translations of full predicate names and names for inverse properties (see Def. 1) are based on hand crafted list of about 150 items.

Since it is not a very big effort to make such translation for a specific domain, e.g. eLibrary, this could be a good solution even for a final product. Nevertheless we decided to create an algorithm, based on some common predicates naming templates. It transforms the original name to a form that is much closer to natural language. The algorithm defines rules for constructing names of invert properties based on the original name of the predicate.

Transformed predicates are easy to understand and to use even by people without computer background. Transformation process also hides the namespace of each predicate so the users are not confused about its role and meaning.

The verification process of the algorithm, however, is still not finished. Evaluation of the algorithm itself is out of the scope of this paper.

6 Reference Implementation of Accessible Browsing

Based on motivation scenario (see Sec. 1.1), in previous sections we have identified the most important aspects of faceted browsing: a set of basic operations (see Sec. 2), access to context and history of browsing (see Sec. 3), collaborative browsing (see Sec. 4), and adaptability of browsing interface (see Sec. 5).

These component solutions influenced building the MultiBeeBrowse system according to SOA (Service Oriented Architecture) paradigm [3], coupled with AJAX-based user interface. MultiBeeBrowse allows to browse unstructured metadata represented as an RDF graph.

In this section we will present the REST-based services in MultiBeeBrowse SOA, and briefly recap user interface component. We will discuss rationale and main design properties behind this solution.

Why REST solution? As research [17] shows, SOAP is inadequate for implementing web services for Semantic Web applications. An argument for REST [16], in the context of the MultiBeeBrowse service, is that GET action defines an idempotent request, i.e., subsequent calls of the same URL should return the same results. This can ensure that user will get the same results, each time given URL is called. In MBB it is vital for collaborative browsing paradigm (see Sec. 4), and handling history of results (see Sec. 3).

Our goal was also to construct a meaningful URL representing single browse operations, as well as, whole chain of operations building up a browsing query. This would allow advanced users to quickly construct their queries, directly in the web browser address field.

Overview of MBB services We have identified following types of services that should be delivered by our SOA:

- *browsing services*: access to a resource, search, filter, browse, similar, combine;
- *context and history management services*;
- *meta-services*, which provides access to statistical information, and allows to format response in desired metadata language format;

Each service is specified using BNF notation[15]; it is enough, since most of the services except for the context services provide only GET method implementation. Each BNF specification has been translated into a regular expression. All services has been grouped in a hierarchical structure. When an HTTP request the URL is processed of the request is decomposed (see Fig. 6) into a chain of atomic service calls, by traversing the tree of hierarchy of services. This approach allows to validate the service call before it is actually executed. Plus, it adheres to web approach to deliver as much as it is possible, against existing problems. In the latter case, if one of atomic browse operations is misspelled the service is executed until the last correct one.

Browsing services specified in section 2, deliver the primary functionality of the MultiBeeBrowse component.

- *Access resource service* allows to load metadata about a resource with given URL for further browsing; this service can only be called as the first one in the chain of browsing operations.
- *Search services*, with three implementations: *keywords and advanced search*, *natural language query templates* [9], and *direct RDF query service*. The two first ones were already described in section 2.1; the third one allows to specify query in one of RDF query languages.
- *Filter service* allows to specify selection filter using either predefined names of properties or using full URIs for both properties and values.
- *Similar service* allows to find resource similar to those in given set,

[15] http://wiki.s3b.corrib.org/MBB/SOA

Fig. 6. Example of decomposing browsing query into browsing and meta-service

- *Related service* allows to find resource that are related to the given ones with given property, e.g., (list of publications) → `has creator` → (list of co-authors).
- *Combination service* performs four operations: conjunction, sum, difference, binding, on two given sets of results; it has to be called as the first one in the chain, and takes as a parameters IDs (see *Meta-services*) of two other browsing operations.

Context and history management services The information on the context of browsing is kept in the RDF storage according to a simple ontology. This ontology defines a *Browsing Context* as a set of current browsing queries (*Browse Action*). Each browsing query invokes a *Call* to a MBB service. The distinction between browsing queries and calls to MBB is important to keep track of user actions, through *Browse Action*, and still have single *Calls* independent of the user; this can be used for, e.g., caching purposes later. With context services, users can traverse current browsing contexts, or retrieve all their browsing contexts with the given call.

Meta-services Meta-services allow to render results in one of RDF serializations (*GetRawMetadata*), or in one of feed (RSS, Atom) formats (*GetFeedMetadata*). For adaptable user interface purposes a special meta-service (*GetStatsMetadata*) generates statistics on properties and values a user can select from. Similar meta-service generates a list of most frequently used concepts (*GetFreqConcepts*). Another meta-service, *GetChainMetadata* renders the definition of current chain of browsing operations in HTML or XML. Finally, the *GetId* service generates a unique ID for given browsing query.

7 Evaluation

Since the faceted navigation itself is pretty well evaluated concept [12] we decided to compare our prototype with other somehow similar solutions such as BrowseRDF[16] and Longwell. Second part of our evaluation is dedicated to gain some opinions about each particular solution delivered by MultiBeeBrowse; these solutions did not exist in either of compared systems (BrowseRDF, Longwell).

Initial questions Our survey involved 20 people. Distributions of age and education level are shown on the Fig. 7 and Fig. 8. No one way familiar with either the dataset used in the evaluation or with MultiBeeBrowse.

[16] http://browserdf.org

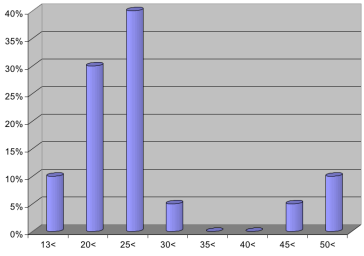


Fig. 7. Distribution of age

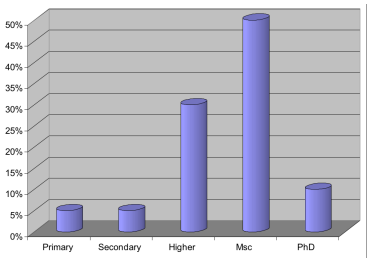


Fig. 8. Distribution of education

First of we asked couple of questions about subjects' background and knowledge of the area of our research. Slightly more than a half of (60%) have computer science background but only 10 of them knew what RDF is. 15 (75%) of subjects were **not** familiar with the term "faceted navigation", and more than 85% use Google to search information on the Web.

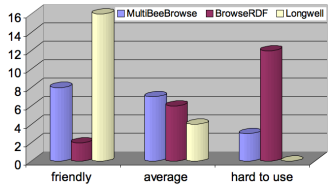
We asked two open, general questions. First was to name some typical tasks they perform during searching. Many (75%) subjects pointed that they simply try to assort good keywords. Then they look on couple first results (30%) or browse the results further (25%). Opening of the results in new browser tabs is also quite popular (20%). Among answers mentioned above we identified: bookmarking, filtering and finding the queries and access to multilevel history of browsing. Second task was to name features subjects miss searching systems they use. Apparently only 20% is fully satisfied - 4 people answered "nothing". In general subjects complained about too many irrelevant results (low precision), lack of quick previews of the results, no possibility of querying in natural language, lack of good history and way to save the results, and last but not least, about problems with too complicated usage and lack of easy help.

We have asked additional pre-assessment questions. And they were (number of positive answers in brackets):

- Have you ever wished you could share your browsing experience with your friends? (45%)
- Have you ever wished you could have someone else help you with browsing? (80%)
- Have you ever wished you could recall you previous search/browse/refinements history? (85%)

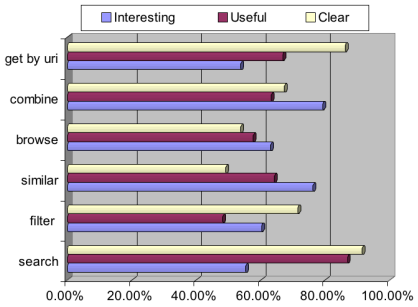
Comparison. We gave a short tutorial showing features of every search system as best as it is possible. All of them were working on the same dataset, taken from http://notitio.us service, filled with FOAF [2] profiles of the users, information about publications from JeromeDL [8] (http://library.deri.ie) and an informal knowledge gathered by IKHarvester[17]. Subjects were allowed to ask questions and try to play with each system before the evaluation.

[17] http://notitio.us/ikh/



	friendly	average	hard to use
MBB	8	7	3
BrowseRDF	2	6	12
Longwell	16	4	0

Fig. 9. Comparison of user friendliness



Type of query	Interesting	Useful	Clear
get by uri	53.8%	66.9%	86.2%
combine	79.2%	63.3%	67.3%
browse	63.1%	57.7%	53.8%
similar	76.2%	64.2%	49.2%
filter	60.3%	48.2%	71.5%
search	55.4%	86.9%	91.5%

Fig. 10. Evaluation of accessibility of browse services

After the presentation, we asked everyone to give a subjective marks (1-10) to each of the compared systems, based on how it looks, how it interacts with the user, features and results it provides.

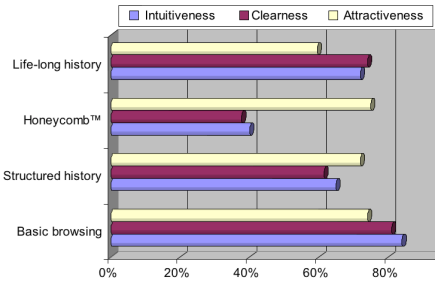
Longwell took first place with average mark 6.875, second (by a proverbial hair's breadth) was our MBB with average 6.8, and left BrowseRDF far behind (avg. 3.8). Since Longwell is a very mature product we found these results satisfying.

We also asked about user friendliness (see Fig. 9), and MultiBeeBrowse came close to Longwell again, leaving BrowseRDF behind.

MBB features In this paragraph we want to show how users value experience our system. Figures 10 and 11 present marks which were given by the users to particular solutions and to parts of our prototype (all marks are in percents).

We also asked some open questions, and the results are very good for us. 18 subjects (90%), like the idea of using results for refining queries. The same number like the idea of saving queries as bookmarks. 15 subjects (75%) can imagine using their friends bookmarked queries for their own searches and vice versa. In particular: while searching for resources (11/55%), helping friends with constructing queries (5/25%) or simply sharing knowledge (12/60%).

As far as user friendliness goes 19 people (95%!) said that the type of the search result was to easy identify, and 17 (85%) of them thought that the default

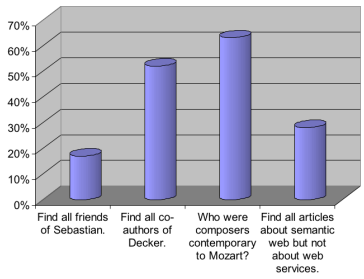


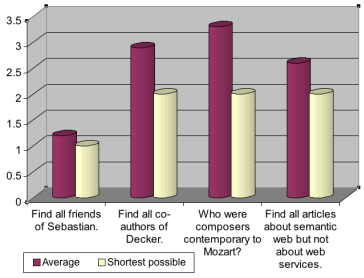
Intuitiveness	Clearness	Attractiveness
72%	74%	59.5%
40%	37.75%	75%
65%	61.5%	72%
84%	81%	74%

Fig. 11. Evaluation of MBB context zooming views

information describing each resource was sufficient. 14 subjects (70%) claimed that the history of searching was easy to identify and 13 (65%) were able to locate themselves while browsing the history. 18 people (90%) found it easy to switch between system components/views.

Task	Confusion	Operations	Time
1	17%	1.2 (1)	24s (10s)
2	52%	2.9 (2)	66s (30s)
3	63%	3.3 (2)	106s (30s)
4	28%	2.6 (2)	51s (20s)





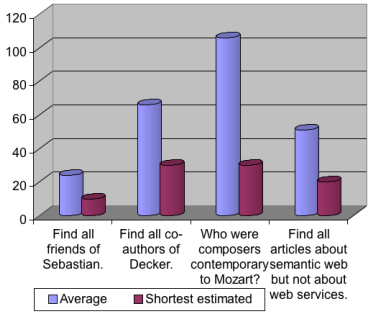


Fig. 12. Evaluation of MBB while performing predefined tasks (confusion, number of operations, time of executing each task)

To finish our survey we gave four tasks/questions to each subject. Solutions to similar tasks with use of all three tools were presented during our short tutorial. The tasks were to find: (1) all friends of Sebastian; (2) all co-authors of Decker; (3) who were composers contemporary to Mozart? (4) all articles about semantic web but not about web services. Subjects were asked about their "level of confusion" while the time and the number of the operations needed were measured. Figure 12 shows the average results (the "ideal" results in brackets). By "ideal" we mean the best result achieved during our pre-survey trials. An average number of operations was pretty close to ideal so the system is not really hard to learn and use and the time was approximately twice longer probably due to lack of user experience.

Comparison between MultiBeeBrowse and other navigation solutions We have also compared features provided by MBB, Longwell, BrowseRDF, and other faceted navigation solutions. From the plethora of services provided by MBB (see Tab. 1), only selection services were fully supported by BrowseRDF, and similarity by Longwell.

8 Conclusions and Future Work

In this article we have presented a concept of an adaptable collaborative browsing interface. We have analyzed operations expected from a browsing solution on interconnected metadata (semantic). We have exemplified how context information can support a history of browsing actions; than we have presented the idea of collaborative browsing. Finally, we have described adaptive solutions for the user interface, and presented the REST-based SOA, implemented in our prototype called MultiBeeBrowse. We have also analyzed evaluation results.

MultiBeeBrowse component has been successfully implemented in notitio.us; it is a service for collaborative knowledge aggregation and sharing; it employs IKHarvester for retrieving RDF information about web resources bookmarked by the users. In contrary to other bookmarking services, such as del.icio.us, notitio.us keeps rich, semantically interconnected metadata shared by the users using Social Semantic Collaborative Filtering [7]. MultiBeeBrowse has also been integrated into JeromeDL 2.1, where together with solutions like Exhibit and TagsTreeMaps it enhances user browsing experience.

Table 1. Comparison of browse operations supported by different solutions

Operator	MMB	BrowseRDF	Longwell	Flamenco,mSpace,Ontogator,Spectacle,Seamark
search	+	±	±	-
selection	+	+	±	±
property	+	±	-	-
browse	+	-	-	-
combine	+	±	±	±

During our work, and further evaluation of MultiBeeBrowse, we have identified a number of goal for future development of the project. These include AJAX proxy service, for aggregating large number of service calls from the user interface.

References

1. P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User Adapted Interaction*, 6(2-3):87–129, 1996.
2. L. Dodds. An Introduction to FOAF. http://www.xml.com/pub/a/2004/02/04/foaf.html, February 2004.
3. T. Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall Professional Technical Reference, 2004.
4. A. Harth, S. R. Kruk, and S. Decker. Graphical representation of rdf queries. In L. Carr, D. D. Roure, A. Iyengar, C. A. Goble, and M. Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006*, pages 859–860. ACM, 2006.
5. M. Hildebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*, pages 272–285, 2006.
6. E. Hyvonen, S. Saarela, and K. Viljanen. Ontogator: Combining view- and ontology-based search with semantic browsing. In *In Proc. of XML*, 2003.
7. S. R. Kruk and S. Decker. Social Semantic Collaborative Filtering with FOAF-Realm. In *Semantic Desktop Workshop, ISWC 2005*, 2005.
8. S. R. Kruk, S. Decker, and L. Zieborak. JeromeDL - Adding Semantic Web Technologies to Digital Libraries. In *Proceedings of DEXA'2005 Conference*, 2005.
9. S. R. Kruk, K. Samp, C. O'Nuallain, B. Davis, B. McDaniel, and S. Grzonkowski. Search interface based on natural language query templates. In *Proceedings of IADIS International Conference WWW/Internet 2006*, 2006.
10. F. Manola and E. Miller. RDF primer. W3C Recommendation, 2003.
11. J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. Number ISBN 156205810X. New Readers Publishing, 2001.
12. E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for RDF data. In *Proceedings of ISWC'2006*, 2006.
13. S. R. Ranganathan. Hidden roots of classification. *Information Storage and Retrieval*, 3(4):399–410, 1967.
14. M. Schraefel, M. Wilson, A. Russell, and D. A. Smith. mspace: Improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM*, 49(4), 2006.
15. V. Sinha and D. R. Karger. Magnet: supporting navigation in semistructured data environments. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 97–106, New York, NY, USA, 2005. ACM Press.
16. S. Vinoski. Putting the "web" into web services. web services interaction models. *IEEE Internet Computing*, 6(4):90–92, 2002.
17. M. Wozniak. Service oriented architecture for collaboration and negotiation ontology management portal. Master's thesis, Gdansk Univeristy of Technology, Poland, 2006.
18. P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *Proceedings of ACM CHI 2003*, 2003.