Semantic Web 0 (0) 1
IOS Press

Exploratory querying of SPARQL endpoints in space and time

Simon Scheider [b,*], Auriol Degbelo [a] Rob Lemmens [c] Corné van Elzakker [c] Peter Zimmerhof [a]
Nemanja Kostic [a] Jim Jones [a] Gautam Banhatti [a]
[a] *University of Münster, Institute for Geoinformatics, Münster, DE*
E-mail: {degbelo, peterzimmerhof, kostic_nemanja, jim.jones, g_banh02}@uni-muenster.de
[b] *ETH Zürich, Inst. f. Kartografie u. Geoinformation, Zürich, CH*
E-mail: sscheider@ethz.ch
[c] *University of Twente, Faculty of Geo-Information Science and Earth Observation (ITC), Enschede, NL*
E-mail: {r.l.g.lemmens, c.vanelzakker}@utwente.nl

Abstract. The linked data Web provides a simple and flexible way of accessing information resources in a self-descriptive format. This offers a realistic chance of perforating existing data silos. However, in order to do so, space, time and other semantic concepts need to function as dimensions for effectively exploring, querying and filtering contents. While triple stores, SPARQL endpoints, and RDF were designed for machine access, large burdens are still placed on a user to simultaneously explore and query the contents of a given endpoint according to these dimensions. First, one has to know the semantic concepts and the type of knowledge contained in an endpoint a-priori in order to query content effectively. Second, one has to be able to write and understand SPARQL and RDF. And third, one has to understand complex data type literals for space and time. In this article, we propose a way to deal with these challenges by interactive visual query construction, i.e., by letting query results feedback into both exploration and filtering, and thus enabling exploratory querying. We propose design principles for SPEX (Spatio-temporal content explorer), a tool which helps people unfamiliar with the content of SPARQL endpoints or their syntax to explore the latter in space and time. We test these principles in a user study on a repository of historical maps.

Keywords: SPARQL endpoints, visual query formulation, spatial and temporal linked data, exploratory querying, historical maps

1. Motivation

Data repositories often remain black boxes, for lay persons and database experts alike and regardless of whether the repository in question is a traditional relational database, a Web-based fast indexing and search system such as Elasticsearch[1], or an RDF[2] triple store[3]. This is surprising, since all this technology is rather mature and was designed for exactly that: helping people retrieve particular information. Linked open data, even though it changes the conditions for and approaches to information retrieval in positive ways [33,21], is not an exception.

Think about an historian or librarian who wants to find maps which contain information about Napoleon's invasion to Russia [28], compare Figure 1. Suppose he or she is given a list of URIs of SPARQL[4] endpoints[5] containing RDF descriptions of large collections of historical maps [20]. How to proceed from there? Even if the SPARQL syntax is known, even if text search and linked data browsers are available, how

*Corresponding author

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

[2] Resource Description Framework, http://www.w3.org/RDF/

[3] See [30] for a short introduction to triple stores.

[4] http://www.w3.org/TR/rdf-sparql-query/

[5] RESTful RDF repository interfaces for sending SPARQL queries on the Web. http://labs.mondeca.com/sparqlendpointsstatus/ is a list of available endpoints.

could the historian possibly know what to tell the endpoint in order to get what he or she wants?

In this article, we suggest that the main challenge in answering these questions is a certain dilemma which has to do with the lacking integration of exploration and querying, and correspondingly, with the lacking possibility of *simultaneously learning about the information needed as well as specifying it*. There are three important issues: First, text based search is too imprecise. Second, knowledge about data schemas and content is required for effective queries but is, in most cases, not available. And third, the syntaxes of data format (e.g. RDF) and query languages (e.g. SPARQL) require expert knowledge.

Piecewise data exploration (e.g. faceted browsing) is not enough in this context because it does not allow users to perform complex searches and does not scale across larger datasets which cannot be explored manually. In information retrieval and database technology, there are two traditional kinds of counter measures: either, one ignores data concepts and exploits text search, or one makes use of data concepts and formulates queries. From a user's viewpoint, text search is simple because it does not require any a-priori knowledge, however, it is also imprecise. For example, if the historian wants to find a map about Napoleon's invasion to Russia, searching for "Napoleon" or "Russia" will not lead to any success, because the intended map (cf. Figure 1) happens to have the title "Carte Figurative des pertes successives en hommes de l'armé française dans la campagne de Russie 1812-1813". A database query may identify the right map because it works independently from text labels, however, it remains unclear, first of all, how to formulate the right query, and second, which kinds of relations and concepts are required and available in order to formulate the right query [28]. Furthermore, in our case, the historian might want to filter results by a certain spatial area and a certain time interval. For this purpose, he or she may need to handle complex geometric data models (since regions can have complex polygon shapes) as used in Geographic Information Systems (GIS) to perform the query.

Systems which make strong assumptions about the technical skills and knowledge of users cannot be used by most of them. On the other hand, a query system that allows lay users to acquire such knowledge needs to support them in *interacting* with concepts and data to some extent. That is, at the end of the day, successful queries require exploration and scalable exploration requires queries. While similar insights have recently motivated *exploratory query* [17] and *exploratory search* [24,34] as fields beyond information retrieval, it remains an open question how Semantic Web technology and how space and time can be used in order to realize it.

Much is being written about linked data visualization tools nowadays, however, we experience a lack of systematic and empirically tested design principles for tools that combine query with exploration on the three dimensions of space, time and theme[6]. The few empirical studies in the field are rather focused on visualization and exploration of linked data as such [12,6,10,2].

In the remainder, we discuss corresponding challenges in detail and suggest a number of design principles as counter measures. We then introduce a SPARQL query and exploration tool which is based on these principles and which serves to test them based on a usability study on finding maps in a historical map repository. In our lab, we analyzed usability during the performance of a number of map retrieval tasks.

2. Joining linked data exploration and retrieval

Joining exploration with querying parallels the old dilemma of joining the learning of concepts with their specification: How to specify something if that something is basically unknown? This dilemma may have stimulated the relational schema in classical database research. A relational schema [9] allows users to specify a piece of data without knowing all the details (e.g. its particular value). However, for this purpose, users need to explore the schema. Note that *exploration*, in our sense, is something that people do when looking at and manipulating a display in order to learn about something, e.g., when looking up a table schema in a relational database. *Specification* is something that people do when they construct retrieval requests, e.g., by specifying a "select" query. *Retrieval* is something that only machines do. However, it often requires specification as input and may have exploration applied to its output.

Since linked data is inherently self-descriptive, it can be used to do both specification and learning about concepts at the same time. Technically, *linked metadata are simply linked data*, and as such are not hidden in the data structure [21], in contrast to a relational database schema[7]. This principle allows users to learn

[6] What we have in mind here are studies like [22].

[7] Syntactically, table schemas are different from table contents and thus need special syntax for look up.

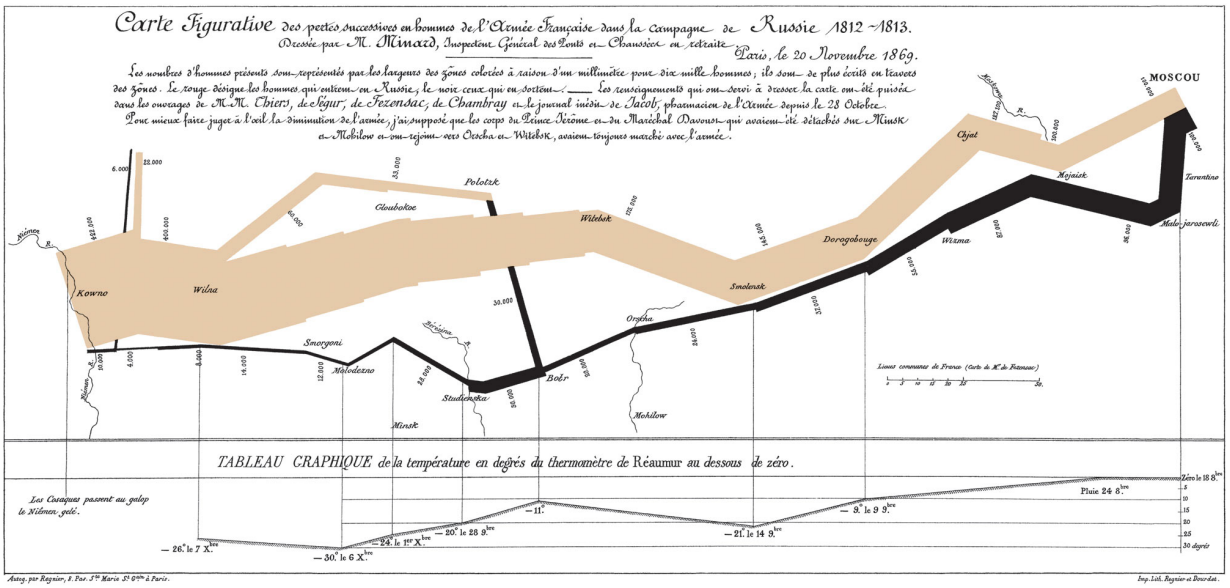

Fig. 1. Charles Minard's map from 1861 about Napoleon's 1812 march on Moscow. It is impossible to search for this map without knowing what it contains.

about data schemas (ontologies) *independently* from retrieval systems [27].

Users can therefore perform exploration and retrieval in a *closed loop* (see Figure 2): Finding meta-data is simply done in terms of a certain kind of query (a query for classes and relations, called *classify* here). This information is suggested in a display and can be explored. Exploration feeds back into the construction of an instantiation query (e.g. a query for instances of classes and relations), which retrieves explorable results that can feedback into another query that filters results based on particular instances, from which one can start browsing of instance neighborhoods, and so on.

In the following, we discuss this challenge in greater detail and review the state of the art. Different retrieval strategies (classify, instantiate, browse, text search) and exploration strategies are discussed in the next two sections. Then we discuss query construction, result display and feedback strategies, as well as the opportunities of integrating space and time into a query.

2.1. Retrieval strategies

Which retrieval strategies support users in learning (compare Figure 2)? As Catarci et al. [7] wrote in 1996, one essential kind of interaction with query systems is *to understand how the domain of interest is represented*. Catarci et al. reviewed general approaches to

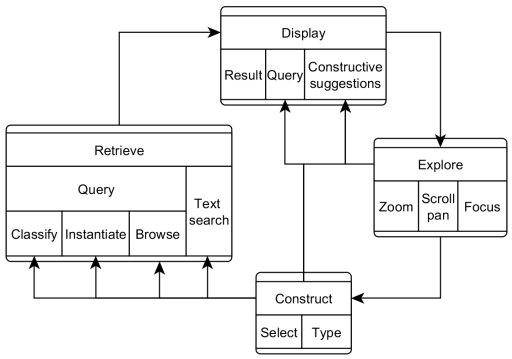

Fig. 2. Schema of chaining exploration and retrieval strategies via displays and query construction. With linked data, this loop can be performed both on meta-data as well as data.

this challenge and distinguish top-down (moving from general to specific) from browsing strategies (moving along neighborhoods of data as well as concepts). In the Semantic Web, top-down query strategies are either instantiations of concepts (finding all instances that satisfy an RDF class description) or browsing of hierarchies of RDF classes and RDF properties. Browsing can also be done on an instance level (browsing is simply a special case of querying, namely querying for neighborhoods of resources in an RDF graph).

In Figure 2, we also added the bottom up query strategy "classify" (finding classes or properties for certain instances, including the entire domain of instances), as

well as direct text-based search of concepts (which is done in current Web search engines).

2.2. Exploration strategies

In parallel to this, Shneiderman [31] distinguished a number of exploration tasks (not only for visualization purposes), namely:

1. overview of the collection
2. zoom in on items of interest
3. details-on-demand for items
4. view relationships on items

Zoom, overview, inspection of details and relationships can be realized by focusing attention on parts of a display, scrolling, panning and zooming in our schema (Figure 2). Note that for all of these exploration tasks, one can choose a top-down, bottom up, browsing or a text retrieval strategy, and for a certain retrieval strategy, one can choose any of the exploration strategies mentioned above. For example, one can search for overview concepts based on their text label or browse through them or query for them. And one can get an overview or visually "zoom into" a concept hierarchy which was itself retrieved by text search or a browser. Retrieval strategies are therefore orthogonal to Shneiderman's exploration tasks.

Since the linked data structure is a graph, graph based visualization techniques seem obvious to address exploration. However, as Schraefel and Karger argue [29], *great big graphs* have severe drawbacks for visualizing linked data sets because what they exactly not provide are overview, zoom and details-on-demand. Instead, a mix of graphical options which fit these tasks and corresponding data types are more adequate. Currently discussed techniques in the Semantic Web range from menus, treemaps and sitemaps or simple lists, to facets and pivoting (concatenating facets) [6]. Visual representations for concepts used in queries can range from table forms, over diagrams (graphs, maps) to icons [7]. One major challenge is to combine those elements on-the-fly to fit a certain purpose or data set [29]. In this paper, we address the challenge of exploring spatio-temporal knowledge encoded in linked data, namely resources and typed literals for space and time, in a simple, cross-domain and user friendly way.

2.3. Query construction strategies

Dadzie and Rowe [10] argued that besides browsing, *basic to advanced query support* is needed to visualize linked data. A visual query system needs to support query formulation [7], and people query data by selecting visual data representation elements and manipulating them. Each possible manipulation needs to translate into a syntactical operation in the formal query language. Therefore, the choice of a visual query language is primarily dependent on the formal query language that needs to be supported ("visual formalism") [7]. In our case, this formal language is SPARQL. Which visual query language should be used for this purpose?

While *faceted browsing* is useful to learn about concepts and data, it is very restricted in expressive power regarding SPARQL. One basically moves a local focus on a node and its immediate property neighborhood from one node to another, selecting values for property ranges. This is well suited for exploration of neighborhoods, but not expressive enough for arbitrary data retrieval. The idea of *pivoting*, i.e., of concatenating facets, allows users to formulate more complex queries, because one can restart a faceted search from the results of another search [6]. However, according to recent results, users have difficulties of using pivoting to build concatenations of relative clauses, like "European countries where an actor has been born who has acted in a Woody Allen film" [6].

Visual or text-based query languages afford the formulation of expressive queries either by *typing text* or by *selecting* visual objects (Figure 2). In general, [7] distinguished different *visual query formulation strategies*, ranging from the (1) *construction of schematic paths (and graphs)* (either by following concept relations or by linking unrelated concepts), over the (2) *composition of concepts and subqueries*, and the (3) *query by examples*, to the (4) *selection of quality ranges*. So far, in the Semantic Web, researchers have focused on visual query languages that help users unfamiliar with SPARQL to generate SPARQL code which allow graph pattern construction, composition and filtering (see for example [35,25,32,8]). vSPARQL [32] represents bound and unbound variables, constants and properties (binary predicates) as edges in a visual graph pattern, which can be visually manipulated, grouped and filtered to generate SPARQL. NITELIGHT uses an ontology browser for selection of classes and properties. GQL and ViziQuer [3,35] allow users to construct queries with an UML based interface. However, all these interfaces primarily focus on SPARQL formulation, less on data exploration, and they do not handle space time literals (compare our discussion in Section 5.1).

An alternative to visual query formulation is the interaction with a restricted set of natural language constructs [11]. This can be very helpful to communicate the meaning of queries, for example of a triple pattern in terms of relative clauses, because it directly translates into a question equivalent to natural language. The challenge here lies in automating high quality translations of patterns and filters that are often dependent on the context of speech.

2.4. Feedback strategies

In formulating a query, search or exploration techniques serve as a starting point to learn about contents. However, during query construction, both techniques can also be used to constantly *map out the space of meaningful query construction steps*. For example, the space of meaningful predicates to choose from reduces dramatically if a subject or object resource is fixed. The current query restricts the possible ways how to proceed, and this can be exploited in *suggestion tools* (Figure 2) that allow a user to map out the space of possible actions. For this purpose, the query needs to feed back into display and exploration tools. And furthermore, query results need to feed back into query construction. In this way, the cognitive load for a user can be immensely reduced, because he or she can query and explore a given repository in a tightly closed loop: Exploration is constrained by a query and vice versa.

2.5. Handling of special datatypes, like space and time, and of URI

The insight of [29] was that every kind of linked data needs a mix of appropriate visualization tools. One way to handle this is to exploit the different kinds of semantics that linked data offers. One can treat the semantics of complex data types (RDF typed literals) with special visualization and query panes, while treating ordinary RDF semantics in a different way.

Complex data types for geospatial data need special attention. There is still a large technical gap between GIS technology, which are based on traditional layer and geometry oriented data models, and linked data [21,16,14]. Developing tools for linked geospatial data is a research task [15,27]. So far, we know of only one data type standard and triple store technology which allows handling of geospatial data on a level comparable to a GIS [4]. A few exploration tools for linked geospatial data have been proposed [1,5,19,18], but we do not know of any tool that exploits the use of maps and time sliders in supporting SPARQL query formulation.

There is also a need to hide the complexity of RDF resources. Users are not interested and do not need to know that resources are identified via URI, because they use ordinary names for this purpose, except when following hyperlinks. Furthermore, if names are not available, then at least RDF namespaces should be handled automatically [8].

3. Design principles

Users unfamiliar with a linked data repository should be supported in finding complex patterns of information of interest about some subject which is hidden in the repository. For instance, users may search through particular buildings on a University campus hosting certain institutes, departments or people. Or they are interested in historical maps of African history from the 19th century which depict the kingdom of Egypt.

Mandel [23, chapter 5] suggested three general interface design principles which can be adapted to the task of supporting query manipulation as follows:

1. The interface should place users into control ..
   - Offer users query manipulation overview, allowing them to navigate through a query and to undo each step.
   - Immediately feed-back results into all displays. In this way, users can explore and learn the function of all manipulations based on feedback.
   - Avoid any menus which could change a display modus (*modeless design*) and instead use context menus and (mouse) foci for exploration.
   - Automatically parse datatypes into appropriate displays.
2. ..it should reduce their memory load ..
   - Always suggest (only) the local space of meaningful query construction possibilities (see explanation in Section 3.2).
   - Progressively disclose options.
   - Use visual cues for possible actions.
3. ..and it should be consistent.
   - Be predictable and exploratory.
   - Use a consistent symbology.

Based on these general objectives, we suggest the following design principles for a visual spatio-temporal query interface for linked data.

3.1. *The syntax is in the visual construction of a query pattern*

Syntactical rules for SPARQL need to be enforced by visual construction. This means that bad syntax is avoided upfront by offering only those construction possibilities that correspond to a valid SPARQL query. Furthermore, every construction possibility needs to be visually indicated and needs to be explained. Users should be able to redo every step and reload a query pattern (a state of construction). Visual cues (e.g. colors) should identify every kind of syntactical element of the SPARQL graph pattern in a consistent manner (variable nodes, constant nodes, property edges, subpatterns...). Furthermore, users should be able to *navigate and focus on parts of a query* in order to modify it. Selection of suggestions should be possible via context menus locally on selected foci.

3.2. *Constructive suggestions are grounded in non-empty results*

Constructive suggestions should always entail a meaningful (non-empty) query result. Otherwise, they should not be displayed. That is, a construction step should not only ensure correct syntax, it should also guarantee *meaningful queries*. For this purpose one needs a *vocabulary suggester* which takes into account the current part of a query in focus and offers exploration/selection of only those vocabulary or data terms which will produce non-empty query results. In this way, it becomes possible to avoid frustrating query attempts and unnecessary browsing. At the same time, users can immediately experience the coverage of data which is available in a repository.

3.3. *Typed literals (space and time) are hidden and automatically handled by "display-filters"*

Typed literals should be hidden from users and should be handled by *display-filter* diagrams. That is, displaying and filtering results should be allowed in the same diagram. This has the advantage that results can be used for exploration as well as for setting of filters. Spatial geometries should be discovered automatically and visualized in maps, and temporal elements in time sliders. Simple filtering can be performed based on the current extent of the respective windows in time and space. In order to offer users such kind of filtering, variable nodes need to be "spatio-temporally-enabled" if corresponding literals are present.

3.4. *Resources are explorable, i.e. automatically labeled and visually linked across displays*

URIs should never be shown to users and only be used as HTML hyperlinks of result items. Instead, all RDF resources should be displayed by their label, and, if not available, by a prefix abbreviation or a label generated from an URI sub-string. Furthermore, result items should be visually linked across different displays, e.g. by highlighting of mouse events, in order to allow users identifying them across displays.

3.5. *Resources feedback into the construction*

Results should be reused in the construction. That is, people should be able to use results directly to filter or restrict a query. This reduces the memory load for users and furthermore allows them to quickly generate smaller result sets which can be visually explored more efficiently.

4. Spatio-temporal Content Explorer (SPEX)

The design principles discussed above were implemented in SPEX, a Web client written in Javascript. We used object-oriented-programming techniques in Javascript in order to divide the code into meaningful parts. Furthermore, we used evolutionary prototyping in order to quickly build a running system with the principles implemented in a preliminary way. We refined this prototype and added further functionalities, depending on user tests. We used the *model-view-controller* pattern [13] in order to separate the logic of the application from the display of data, and the latter two from the interactions with the user. Figure 3 shows the architecture of SPEX. Classes belonging to the *model* component of SPEX are: FilterExpander, LabelGenerator, LiteralExpander, QueryExecutor, Spex, SPEXResultSet, SPEXQuery and Suggester. The *controller* component comprises: FilterPane, QueryPane, Slider and SpaceFilterPane. ResultsPane is the only class belonging to the *view* component of the application. The code of the SPEX is available for exploration and re-use at `https://github.com/lodum/SPEX`. For a working version of the applica-

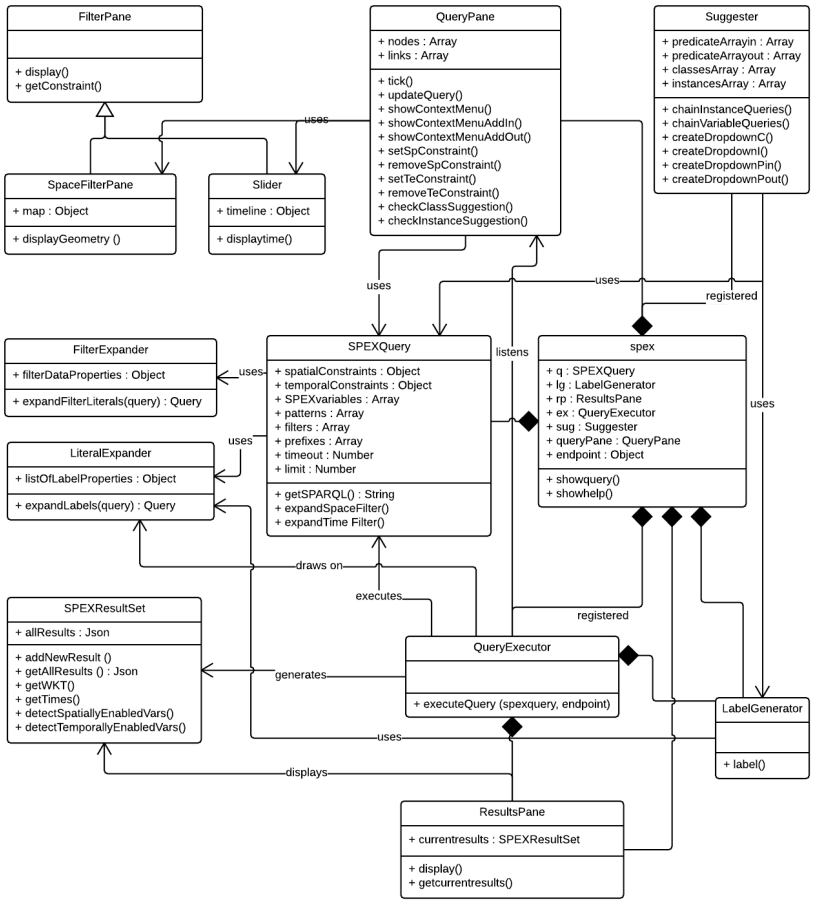

Fig. 3. SPEX architecture. It consists of one *view* (ResultsPane), eight *models* (FilterExpander, LabelGenerator, LiteralExpander, QueryExecutor, Spex, SPEXResultSet, SPEXQuery, Suggester) and four *controllers* (FilterPane, QueryPane, SpaceFilterPane, Slider).

tion, see `http://geographicknowledge.de/spex/`.

The layout of the user interface is shown in Figure 4. The user interface is divided into three window parts, namely the *query pane*, two *display-filters* for space and time and the *results pane*.

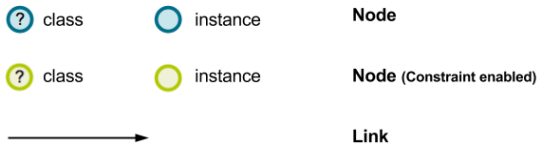

Fig. 5. SPEX visual query constructor elements.

The construction of queries is possible through the use of context menus on nodes (see Figure 6). The query pane is initialized with a single unlabeled variable node, a question mark which stands for an unspecified question. Figure 5 depicts visual query constructor elements of SPEX. There are two kinds of nodes: variable nodes (nodes which represent sets of instances) and instant nodes (nodes which represent

Fig. 6. Context menu on a selected node. It offers node restrictions to things of a kind (i.e. classes) or particular things (i.e. instances), and can be used to specify relationships to/from other things and to carry out space/time filtering on a node.

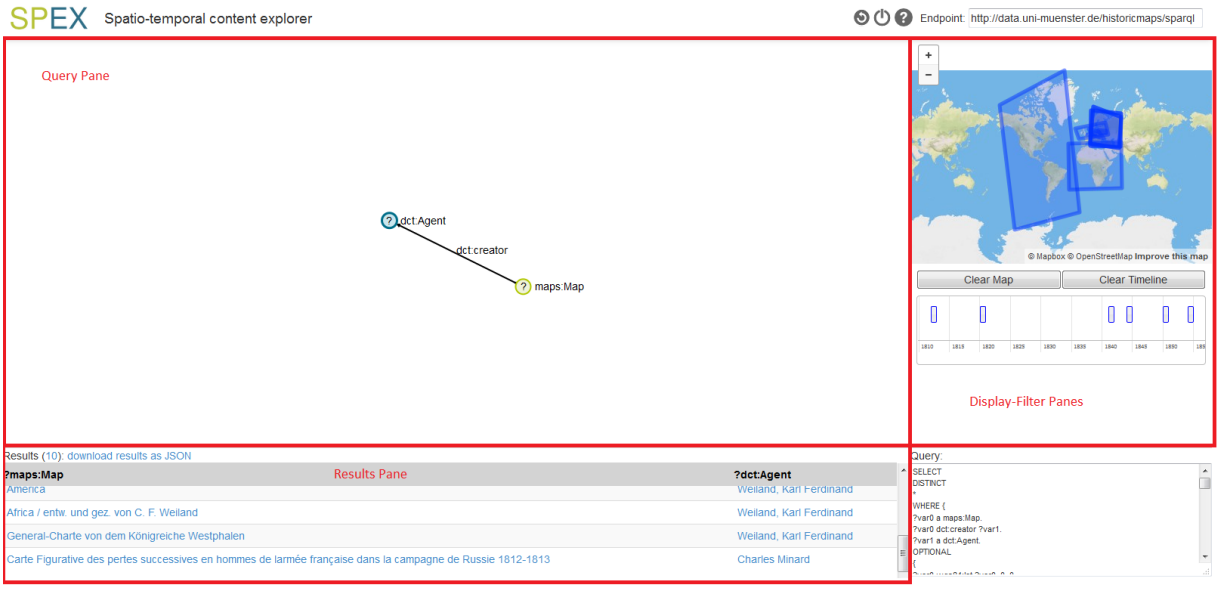

Fig. 4. SPEX layout. It consists of three visual panes (query, results, and display-filters). The SPARQL query generated can be seen on the lower right hand side.

particular instances). The former are identified by a question mark inside the node, the latter are empty. Variable nodes can be restricted to an RDF class and are labeled by the name of that class. Otherwise they are labeled by a variable name. Instance nodes are labeled by the particular resource they denote. Spatial or temporal enabling of nodes is depicted by green vs. blue colors.

A user can add specifics (triple patterns and filters) to a query by selecting an option from a node's context menu (compare Figure 7): (1) either he or she can text search or scroll through available classes or instants connected to this node and submit an instantiation query by pressing return. Class restrictions translate into `?var rdf:type Class` statements, and instance restrictions into filter statements of the form: `FILTER (?var = <URI>)`. Or, (2) he or she can specify in- or out-going relationships (i.e., triple patterns of the form `?var property ?var1`) to the node by text searching and scrolling through a list of available properties connected to the node. Or, (3) he or she can set a spatial or temporal constraint on the node, adding special filters to the query[8]. A spatial or temporal constraint can only be set if a node is spatially/temporally enabled, i.e., it is linked to corresponding literals. The constructive process assures that only valid SPARQL can be generated. Suggester queries running in the background (class *Suggester*) make sure suggested queries are always non-empty. They filter out domain vocabulary RDF terms (including classes, properties and instance names) that are contained in the endpoint and linked to a node, excluding syntactical RDF terms. Result RDF resources are clickable and automatically labeled and displayed in terms of a scrollable table. Furthermore, results are projected to time line events and map window objects that are visually linked to table items (via mouse over highlighting), provided that corresponding literals are present. This allows multidimensional exploration of results (Principles 3.4 and 3.5).

The current development covers only a subset of SPARQL. It allows the construction of *connected conjunctive triple patterns* and focuses on SELECT

[8]Map window and time slider window can be zoomed and used to filter out results based on whether linked literal values (e.g., WKT geometries in GeoSPARQL [4] or time instances/intervals in OWL-Time (`http://www.w3.org/TR/owl-time`)) are contained in a window or not. The spatial filtering is currently done inside the SPEX client in order to remain independent from endpoint technology. Most triple stores to date cannot deal with complex (non-point-like) spatial geometries on the level of the Open Geospatial Consortium (OGC) `http://www.opengeospatial.org/`, such as WKT `http://en.wikipedia.org/wiki/Well-known_text`. In the future, we intend to add server side spatial filtering for performance improvement.

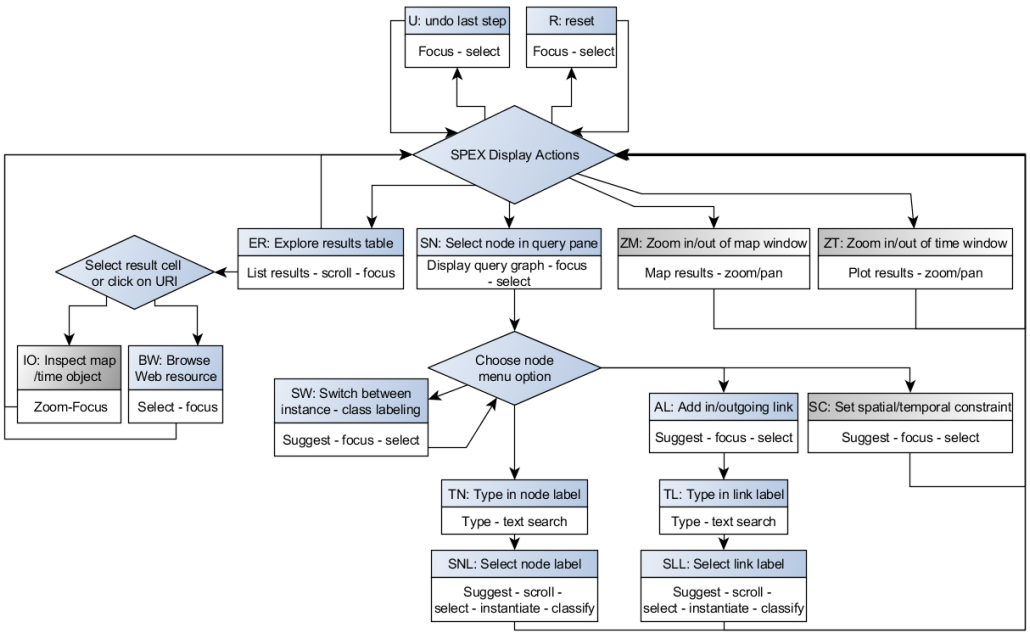

Fig. 7. Flow chart of visual query construction and exploration in SPEX. Each process step is represented by a box and a 2-letter short name and is further subdivided into display - explore - construct and retrieval actions (compare Figure 2). Dark grey boxes denote actions with spatio-temporal displays.

queries with '*', since CONSTRUCT, ASK queries and the distinction of bound and unbound variables are not needed for exploration purposes. It does not yet allow users to deal with *graph patterns*, and thus does not support the formulation of disjunctive (via UNION) and OPTIONAL patterns. This was not required in our scenario (see Section 5.1) but can be added in the future. FILTER expressions are supported in two ways: (i) In terms of display-filters for selected typed literals (space, time), according to our design principle 3.3; or (ii) As query constraints on variables denoting individual resources, which enables turning a variable node into an individual node. Note that our focus was not on expressive power, but on integrating content, space and time exploration with querying for librarians.

While SPEX implements most of our design principles, there is room for improvement by combining different exploration and query strategies. First of all, more sophisticated exploration tools for vocabularies (ontology browsers) could be used, as implemented in [26]. This could be combined with other overview strategies [6]. Second, the display and browsing of results may be turned from a simple table into faceted browsing [6]. However, the current solution supports visual linking of results by highlighting across displays, which is not straightforward in a faceted browser. Lastly, further query construction strategies could be used, such as query by examples [7]. All these options were not essential in our scenario, which is described next.

5. Exploring a repository of historical maps

In this section, we evaluate the functionality of SPEX based on a scenario and a number of corresponding retrieval tasks. We also discuss the design of our user study which was done based on this scenario.

5.1. Scenario, tasks and challenges

We tested our approach based on exploring a SPARQL endpoint[9] about historical maps from the University library in Münster[10]. We asked subjects, all of them lay

[9] http://data.uni-muenster.de/historicmaps/sparql

[10] http://sammlungen.ulb.uni-muenster.de/nav/classification/116654

persons with respect to SPARQL and RDF, to answer a list of questions about this map repository (correct answers given in brackets):

1. Maps with lakes
   (a) How many maps represent lakes? (22)
   (b) List those represented lakes that have a name (Lamak lake, Lac du Castillon, Lac du Chambon, Lac de Serre-Poncon)
2. Maps with cities
   (a) How many historical maps represent cities which are in Poland? (1)
   (b) Show these cities in the browser (“Stettin”)
3. Maps with military conflicts
   (a) How many historical maps represent military conflicts? (2)
   (b) Which years do these maps represent? (1801, 1812)
4. Maps about Africa
   (a) How many maps depict Africa? (5)
   (b) How many of them were created by ”Karl Ferdinand Weiland”? (1)
   (c) Which years do these maps represent (1841)?
5. Maps about Europe in the 18th century
   (a) How many maps show historical periods in the 18th century (6)
   (b) Which one of these has the largest map scale (largest scale ratio or smallest divisor)? (“Gefecht bey Reichenberg in Böhmen”)
   (c) Name a village which is represented in this map (“Reichenberg”)

These questions where chosen such that it is nearly impossible to answer them by simple text search and that it is difficult to answer them by piecewise exploration, but instead require a combination of query and exploration, where queries involve relationship and class instantiation as well as classification. Furthermore, as we discussed in [28], these kinds of questions correspond to knowledge about document content that is (at least in principle) of interest to historians and historical cartographers, but cannot be posed in current library search systems [27]. The questions require handling of space and time at least to some extent. For these reasons, they represent a scenario which is both realistic and challenging for people unfamiliar with SPARQL and the repository.

In SPEX, the tasks can be most efficiently solved by the following procedures (compare process steps in flowchart model of Figure 7):

```
Q 1a ``How many maps represent lakes?'': SN[
    node1]->TN[Map]->SNL[maps:Map]->SN[node
    1]->AL->TL->SLL[maps:mapsPhenomenon]->SN[
    node 2]->TN[Lake]->SNL[phen:Lake]->ER
    ->[``22'']
Q 1b ``List those represented lakes that have
     a name'': ->ER->[``Lamak lake, Lac du
    Castillon, Lac du Chambon, Lac de Serre-
    Poncon'']
Q 2a ``How many maps represent cities which
    are in Poland?'': ->R->SN[node1]->TN[Map
    ]->SNL[maps:Map]->SN[node 1]->AL->TL->SLL
    [maps:mapsPhenomenon]->SN[node 2]->TN[
    City]->SNL[phen:City]->SN[node 2]->AL->TL
    ->SLL[dbp:country]->SN[node2]->SW[
    instance labels]->TN[Poland]->SNL[dbp:
    Poland]->ER->[``1'']
Q 2b ``Show these cities in a browser'': ->BW
    [dbp:Stettin]
Q 3a ``How many historical maps represent
    military conflicts?'': ->R->SN[node1]->TN
    [Map]->SNL[maps:Map]->SN[node 1]->AL->TL
    ->SLL[maps:mapsPhenomenon]->SN[node 2]->
    TN[Conflict]->SNL[phen:MilitaryConflict
    ]->ER[``2'']
Q 3b ``Which years do these maps represent
    ?'':->IO->[``1801, 1812'']
Q 4a ``How many maps depict Africa?'': ->R->
    SN[node1]->TN[Map]->SNL[maps:Map]->SN[
    node 1]->AL->TL->SLL[maps:mapsPhenomenon
    ]->SN[node 2]->SW[instance labels]->TN[
    Africa]->SNL[Africa]->ER[``5'']
            [alternative: using ZM to Africa]
Q 4b ``How many of them were created by Karl
    Ferdinand Weiland?'': ->SN[node 1]->AL->
    TL->SLL[dct:creator]->SN[node 3]->SW[
    instance labels]->TN[Weiland]->SNL[Karl
    Ferdinand Weiland]->ER[``1'']
Q 4c ``Which years do these maps represent
    ?'': ->ZT->[``1841'']
Q 5a ``How many maps show historical periods
    in the 18th century?'': ->R->SN[node1]->
    TN[Map]->SNL[maps:Map]->ZT[1700-1800]->SN
    [node 1]->SC->ER->[``6'']
Q 5b ``Which one of these has the largest
    maps scale?'': ->SN[node 1]->AL->TL[Scale
    ]->SLL[maps:hasScale]->ER->[``Gefecht bey
     Reichenberg in Boehmen'']
Q 5c ``Name a village which is represented in
     this map'': ->SN[node 1]->AL->TL->SLL[
    maps:mapsPhenomenon]->SN[node 2]->TN[
    Village]->SNL[phen:Village]->ER->[``
    Reichenberg'']
```

Listing 1: Flowchart based procedures for expected solutions.

Table 1

Qualitative comparison of SPARQL tools against our principles and scenario challenges

Criterion	NITELIGHT	SPARQLViz	ViziQuer	iSPARQL	Rhizomer	RelFinder	SPEX
Validity of visual construction (3.2)	yes	yes	yes	yes	yes	yes	yes
Vocabulary Suggestion (3.3)	yes	no	yes	no	yes	yes	yes
Suggestion grounded in results (3.3)	no	no	yes	no	yes	yes	yes
Handling of space and time in queries (3.4)	no	no	no	no	no	no	yes
Exploration support (3.5)	(yes)	no	no	no	yes	yes	yes
Results feedback into construction (3.6)	no	no	no	no	yes	no	yes
Target groups	experienced	experienced	experienced	experienced	lay person	lay person	lay person
Query expressivity suitable for scenario	yes	yes	yes	yes	yes	no	yes

5.2. Functional tool comparison

Our test scenario challenges a number of tools offered in the Semantic Web (see Table 1). As discussed in the extensive review of Dadzie et al. [10], text based search tools primarily target tech-users and do not support the exploration and querying of relationships, which is required here.

From the available *visual SPARQL tools*, only NITELIGHT, IsaViz (SPARQLViz), ViziQuer and OpenLink iSPARQL can be directly used as clients on SPARQL endpoints. From these, iSPARQL and SPARQLViz require a-priori knowledge about contained vocabularies for building a meaningful query, and they are primarily made for tech-users. Only ViziQuer and NITELIGHT seem to have a form of overview and suggestion tool for available vocabularies and substantial support in building queries.

From the available *linked data browsers with visualisation options* we found that only LESS, RelFinder, RDF Gravity[11], Tabulator[12] and Rhizomer[13] support SPARQL queries, and from these only RelFinder directly works on SPARQL endpoints. RelFinder can be used without a-priori knowledge and supports autosuggestion, however, it is restricted to instance based queries without variables, and thus does not support any of the questions above.

Table 1 lists a selection of tools and compares them against our principles and scenario challenges. In summary, it seems that current tools are challenged by our scenario because they either lack query expressivity or are designed for experienced users or do not support exploration. Tools which seem to offer comparable functionality according to our principles are Rhizomer and ViziQuer.

ViziQuer has a class and property suggester which is grounded in results, very similar to the one we implemented in SPEX, and thus can prevent users from building non-meaningful queries. However, it has a complex expert-oriented user interface which still allows forming non-meaningful queries, and results are not feed-backed into construction and exploration.

Rhizomer is more than a faceted browser in that it supports expressive queries with variables, properties and classes, supplies only grounded suggestions and is made for exploration of lay persons. However, compared to SPEX, its exploration and query strategies seem less tightly integrated (see Figure 8 and compare to Figure 7), as query results feedback into a query only via facets. Furthermore, in Rhizomer, arbitrary class restrictions require filtering of values of the facet

[11] http://semweb.salzburgresearch.at/apps/rdf-gravity/

[12] http://www.w3.org/2005/ajar/tab

[13] http://rhizomik.net/html/rhizomer/

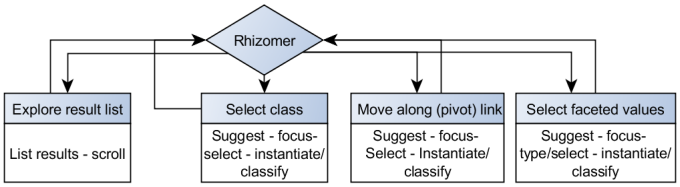

Fig. 8. Flow chart of query and exploration in Rhizomer. In order to build complex queries, users can either select a faceted value (an instance linked via the facet property) or move along a link via pivoting.

"rdf:type", so that a user needs to understand the RDF syntax to some extent. Most importantly, however, Rhizomer's browser and pivoting-based visualisation strategy makes it difficult to get an *visual overview of a query*: One needs to move along a link (pivot) or read a natural language -like description of the current query. In addition, the pivoting option is visually separated from the facets, even though both are based on the same set of links. This makes it difficult to learn how the current query can be extended.

5.3. User study design

We tested the usability of SPEX by observing how users performed on the tasks in Section 5.1. We did not do an empirical comparison of different tools. One reason was that the functional differences between them seemed too big to allow for reasonable quantitative comparisons (compare results in Table 1). Furthermore, we were more interested in a preliminary qualitative study of our own tool in order to discover whether our design principles work and to find the reasons for remaining problems.

The SPEX tool was tested with potential users in two stages: in the first stage a heuristic evaluation was done with an expert. The outcomes of this evaluation were used to adjust and improve the information in the "Help file", the formulation of the tasks (see section 5.1) and some aspects of the interface. Thereafter, the tool was tested with seven subjects (one of them only in a pre-test) who are familiar with the problem of searching for geographic information on the basis of particular criteria. Two subjects are employed as librarian / information specialist in two different universities, two other test persons are currently doing PhD research in the geo domain and the last two test persons are scientists / lecturers at the Faculty ITC of the University of Twente.

The test persons were first asked to familiarize themselves with the SPEX tool through the "Help file" (which was printed on paper) and SPEX displayed on the screen. They were allowed, and even encouraged, to interact with the interface to try to better understand its functioning and design. Thereafter, the test persons were asked to execute the tasks as described in Section 5.1. One test person took 94 minutes to go through the Help file and familiarize himself with the tool. The other subjects took less time, but still 45 minutes or more. For the execution of the tasks, usually 30 to 45 minutes were needed, although not all test persons could complete the tasks.

The use and user issues of the SPEX prototype were discovered through qualitative user research in which a mix of user research methods has been applied. Such a mixed methods approach is quite common in current user research, as different methods lead to different and complementary information about the interaction of a subject with the tool or interface. So, in our research we observed the test persons while they were going through the "Help file" and while they were executing their tasks with SPEX. We did that by asking them to constantly think aloud (audio recorded), by screen and event logging (mouse clicks, etc.), by eye movement registration and by video recording their facial expressions. All these recordings were synchronously made, stored and analyzed in a Tobii X60 hardware and Tobii Studio 3.2.1 software envionment, installed in the cartographic usability laboratory of the Faculty ITC of the University of Twente.

The 8 - 10 hours recordings were analyzed in a qualitative way by matching the SPEX interaction model (Figure 7) with the thinking aloud and activity protocols.

The execution of the user tests went fine although some test persons had difficulties with thinking aloud (they had to be reminded to do so every now and then by the research leader who was in the same room all the time) and the eye movement registration of some subjects sometimes also suffered from the fact that at the end of the long sessions the test persons substantially moved their chairs and bodies. The research leader did not interfere with this in order not to disturb

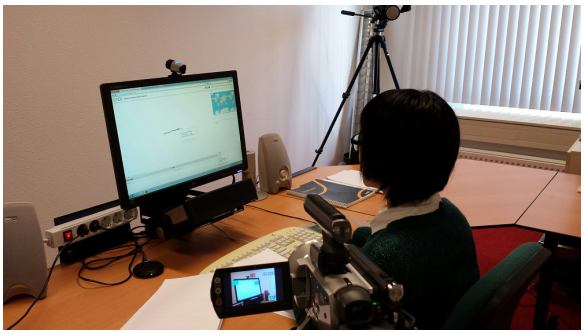

Fig. 9. The configuration of screen, eye tracking and video recording with one test person at ITC.

the think aloud process. All in all, the recordings resulted in a substantial and valuable amount of research data. At the end of each test session, the research leader briefly interviewed each subject and asked them about their general satisfaction with SPEX.

6. Evaluation of results

We did a qualitative evaluation of user tests based on analyzing their thinking aloud utterances and their eye-tracked screen videos. For each user test, we recorded the path they were following in terms of our flowchart from Figure 7 to see which exploration-query steps they were performing, and which of them led to a success regarding our tasks, which didn't, and why.

A synopsis of successes and failures is given in Table 2: "success" means that the question was answered correctly and as expected; "failed" means that the answer and the way to the solution were incorrect; "success (quasi)" means the way of solution was entirely correct but the answer was not, and this only because of the influence of previous failures (dependencies between answers); "partial" means that the question was not answered correctly, but a significant part of the way to the solution was correct; "skipped" means that the question was skipped for some reason (e.g., because it depends on predecessors which were not answered, or because participants were exhausted or simply forgot to give an answer). The addition of "(a)" for "alternative" means that the test person used an alternative (unforeseen) but in principle valid way to the solution. We also added the solution ratio, which is the proportion of successfully answered questions (either success or quasi success) divided by the number of answers given (without skipped questions).

What can be seen from this table can be summarized as follows:

1. SPEX in principle allows lay persons to answer all of these questions about an unknown repository[14]. The most difficult questions, however, require a lot of reasoning and iterative trials by the test persons. Question 5 (the most difficult one) was obviously too difficult for beginners.
2. Complete failures seldom occur. Persons chose at least partially valid ways to an answer (compare test person 1). Some of them even figured out creative ways to a solution that we did not anticipate (see discussion below for examples).
3. Furthermore, taking into account that the difficulty and level of complexity of our questions was considerably high right from the start and increased a lot towards the end, we notice that the level of success nevertheless remained quite high. A considerable exhaustion of users can be observed towards the end in terms of a significant increase of skipped trials. However, there is not an equally steep drop of successes until question 5, which probably shows a rather steep learning curve.
4. The space-time interface was used frequently (Questions 3b, 4b, and 5a). Some people skipped the tasks, but mostly because they thought the solution was obvious after they tried out the map and time slider. The rest solved the questions rather successfully (4 successes and 3 partial solutions from 9 trials). However, we also discovered some usability problems with the space-time interface (see below), and addressing these could improve this result.

Listing 2 shows an excerpt of our flowchart based recordings for test person 1 in Table 2. Explanation text in square brackets was added by us, and quotes were uttered by the test person.

```
Q 3a ``How many historical maps represent
     military conflicts?'': R->SN->SNL[Map]->
     SN->AL->SLL[mapsPhenomenon, ``Military
     conflict is a phenomenon, probably
     '']->[``Strange that those arrows are
     always pointing into a different
     direction'' (visual arrows direction vs.
     text direction)] ->SN[selects phenomenon
     node]->SNL[``Military Conflict'']-> [
     correct answer: 2]
Q 3b ``Which years do these maps represent
     ?'':  ->ER->IO[inspects time events]->[
     correct answer: 1801, 1812]
```

[14]This conclusion is drawn based on the results from the first five test persons as the last test person skipped most of the questions.

Table 2
Successes and failures of test persons in finding answers to questions from Section 5.1

User	Q 1a	Q 1b	Q 2a	Q 2b	Q 3a	Q 3b	Q 4a	Q 4b	Q 4c	Q 5a	Q 5b	Q 5c	sol. rat.
1	success	success	partial	skipped	success	success	skipped	success (quasi)	success (quasi)	partial	skipped	skipped	6/8 (0.75)
2	partial	success	success	success	success	skipped	partial	skipped	skipped	failed	skipped	skipped	4/7 (0.57)
3	partial	success (a)	success	success	partial	skipped	success	success (a)	skipped	partial	failed	skipped	5/9 (0.55)
4	partial	success (quasi)	success (a)	success	success (a)	partial	failed	skipped	skipped	failed	skipped	skipped	4/8 (0.5)
5	partial	partial	success (a)	success	partial	skipped	failed	success (quasi)	failed	skipped	skipped	skipped	3/8 (0.38)
6	partial	success (quasi)	partial	skipped	skipped	skipped	skipped	skipped	skipped	skipped	skipped	skipped	1/3 (0.33)
sol. rat.	1/6 (0.16)	5/6 (0.83)	4/6 (0.66)	4/4 (1.00)	3/5 (0.6)	1/2 (0.5)	1/4 (0.25)	3/3 (1.00)	1/2 (0.5)	0/4 (0)	0/1 (0)		

```
Q 5a ``How many maps show historical periods
    in the 18th century?'': R[restart with
    new window]-> SN->SNL[Maps]->[``And now
    the 18th century. Oh I should query that
    ..'']->SN-AL-SLL[does not find property,
    ``What does time mean?'', select date]->
    ER[``Is date the one?'' ``I don't know
    how to do this. I can do it with the
    timeline. But what does the timeline tell
     me? Is it the creation date?'']-> U[
    undoes; would like to have a delete
    function in the node]->SN->ZT[scrolls
    timeline to 18th century; clicks on
    events and wonders what happens] -> SW[
    confused; ``I am doing too many things in
     one window here'']->AL->SLL[``has a
    period. That's what I am looking for'',
    but only finds date, select data again]->
    ER[``I only get one. There is nothing
    that is called a 'period'.'']
```

Listing 2: Flowchart based recording of test person 1.

Comparing Listing 2 with Listing 1 illustrates how usability problems can be spotted immediately.

In the following, we will discuss the main insights from our analysis of the usability test. First, we list our observations regarding successful user trials and what this tells us about the functioning of our design principles:

- People made effective use of all visual query graph construction possibilities. Choosing node labels and adding visual edges seems in general to translate well into the questions of the task (provided, however, that people understand the meaning of vocabulary terms). All people used visual link construction as well as node labeling to conceptualize the question and to solve their query task. Test person 1 explicitly formulated questions in terms of node-link query patterns, even though linked data principles were unknown to this person.
- People used result exploration, Web browsing as well as map and timeline exploration in order to answer their questions in ways that were not foreseen by us ("alternative solution"). They used many trials to figure out different ways to reach an equivalent goal (and indeed, "many ways lead to Rome"), e.g.: Using map geometry sizes to figure out map scales; using Web link information (dbpedia) to find information about associated times; Using two filters on space and time and the property "hasScale" to answer questions 5a and 5b, which in principle is a correct (but unexpected) answer.
- People ran through many query-exploration cycles iteratively (which is not time consuming because SPEX is modeless, i.e., works without menu hierarchies, and has an undo/reset button). This allowed them to quickly learn something new which they could use in the next iteration, i.e., they could make iterative learning progress. For example, test person 1 figured out that pressing return is necessary after input in his first cycle, and then used this in all subsequent cycles. Another test person figured out (after several unsuccessful trials with browsing map descriptions on the Web) that map scales can also be compared via geometry sizes in the map window, which lead to an alternative solution.

- People explored term meanings using term suggestions and result feedback. For example, test person 1 explored the meaning of the term "Topographic Map" by checking whether results are a subset of "Map".

The following list contains main difficulties users were confronted with when using the current version of the tool on the given tasks. We classified them according to the causes of their confusion or task failure and added possible cures:

- Difficulties with understanding linked data vocabularies and example (historical map) data models
  - People had difficulties in guessing (without any explanation) the conceptual data model underlying our scenario which treats maps as well as their contents as linked nodes. For example, one test person largely failed to understand that in the repository, maps are entities different from lakes and countries to be selected by a separate node and in need of linking to the latter. This issue has probably to do with the particularly unusual way of modeling map contents by linked data.
  - People had difficulties with abstract and unclear categories such as "hasPhenomenon" in the property list. For example, test person 1 subsumed "lake" under "phenomenon" but not "country" in the case of Poland and therefore failed to answer question 2. Testers also had trouble understanding the difference between *dc:date* and the word "period" as used in question 5a), compare Listing 2. This is a consequence of the particular vocabulary terms used and a lack of their explanation in the current version of SPEX. Understanding vocabularies can be improved by adding vocabulary exploration or explanation tools. For example, a simple measure would be to load RDF term comments from the Web and show them to a user whenever a term needs to be selected.
  - Prefixes occurring in front of terms were disturbing (test person 1: "dct: I have no idea what that is"). Vocabulary prefixes should be explained (or removed).
- Difficulties with the context menu options
  - The query terminology used in context menus was difficult to understand. For example, the distinction between things of a kind and particular things (test person 1: "Why do I have to make the decision?") could be removed.
  - People were easily confused about the necessity of pressing return to go to the next step (extra cognitive effort).
  - Node context menu options were not offered in sequential steps but all together at all times. For example, label selection was offered also after a label was selected. This made it hard to think about what step was next (test person 1: "I would not expect to see maps here in text form if I have selected the label map already." "There should be a difference between what you have left behind and what your next task is going to be".)
- Difficulties with space-time exploration and querying.
  - In general, people said that the map window should be more prominent and should be better explained.
  - The meaning of the link between map and time objects and results remains unclear. Do the events on the time line represent what is shown in a map or when the map was created (compare Listing 2)? Links to space time literals are currently not shown to the user. In general, all views should be highlighted in both directions.
  - People find it difficult to decide whether they should treat time and space constraints via filters or via relationships in the query window. They try out both and are usually exhausted when one option fails and do not try the other option any more.
  - Cluttered map and time objects makes space time windows less useful.

What is important to notice here is that the reasons for usability problems were located either outside of the scope of the intended functionality of SPEX (e.g., supporting the understanding of vocabularies and domain models was not a goal of our design), or they were related to implementation choices that are not based on our design principles (compare Section 3). Some of the discovered implementation problems even seem to exist precisely because the principles were not yet implemented with enough consequence: For example, the syntactical choice between class and instance based queries was driven by our understanding of RDF but contradicts principle 3.1, because it is neither ex-

plained nor even necessary. The missing semantic link between results and map/time objects and missing exploration of result name spaces contradict principle 3.4. And showing a menu option also after its use contradicts the principle of progressive disclosure of options (Compare Section 3).

Overall, users said in the interviews that the tool offers an interesting way of searching. Especially the time window was considered a useful part. This fits and supports our observations about the overall frequent and successful use of query and exploration cycles in the user study. For these reasons, we conclude that the study supports the general functioning of our design principles, even though it highlights also clear opportunities for improvement.

7. Conclusion

In current retrieval systems, query and exploration are usually not tightly integrated and do not handle space and time in both query and exploration. This prevents users unfamiliar with a given repository from taking advantage of the self-explanatory power of linked data and the intuitive retrieval possibilities of maps and time sliders in order to find out about interesting repository content by themselves. In this paper, we have proposed design principles for exploratory querying of SPARQL endpoints in space and time. We have made a functional comparison of linked data tools with respect to these principles, implemented them in terms of the SPEX tool and tested them in a usability study.

Results show that our design principles allow tightly closed query and exploration loops, and thus non-expert users to answer challenging questions about an unknown endpoint which they could hardly answer with current linked data query tools. They also show that modeless design and immediate result feedback enable users to iteratively explore query functionality, data content and vocabulary meaning, as well as to find alternative solution paths for a given question. However, we also found that successful question answering requires a better understanding of vocabularies, improved usability of context menus and a better linking of maps/time sliders with result displays.

In the future, we plan to address these usability challenges by corresponding developments as indicated in Section 6. Furthermore, there are options to improve both the variety of query and exploration methods as well as the scalability of SPEX. The latter requires mainly server side support for spatio-temporal queries.

Acknowledgments

This work has been funded by the German Research Foundation (DFG) through the *Linked Data for eScience Services* (LIFE) project (KU 1368/11-1). We thank our project partners, the Münster University Library (ULB), for their constant support of this work. Furthermore, we thank all participants of our user test for letting us peek over their shoulders.

References

[1] Alonen, M., Kauppinen, T., Suominen, O., Hyvönen, E.: Exploring the Linked University Data with visualization tools. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) The Semantic Web: ESWC 2013 Satellite Events. pp. 204–208. Springer, Montpellier, France (2013)

[2] Bartoschek, T., Pape, G., Kray, C., Jones, J., Kauppinen, T.: Gestural interaction with spatiotemporal linked open data (2013)

[3] Barzdins, G., Rikacovs, S., Zviedris, M.: Graphical query language as sparql frontend. In: Local Proceedings of 13th East-European Conference (ADBIS 2009). pp. 93–107 (2009)

[4] Battle, R., Kolas, D.: Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web 3(4), 355–370 (2012)

[5] Becker, C., Bizer, C.: Exploring the geospatial semantic web with dbpedia mobile. J. Web Sem. 7(4), 278–286 (2009)

[6] Brunetti, J.M., García, R., Auer, S.: From overview to facets and pivoting for interactive exploration of semantic web data. Int. J. Semantic Web Inf. Syst. 9(1), 1–20 (2013)

[7] Catarci, T., Costabile, M.F., Levialdi, S., Batini, C.: Visual query systems for databases: A survey. J. Vis. Lang. Comput. 8(2), 215–260 (1997)

[8] Clark, L.: Sparql views: A visual sparql query builder for drupal. In: Polleres, A., Chen, H. (eds.) ISWC Posters and Demos. CEUR Workshop Proceedings, vol. 658. CEUR-WS.org (2010)

[9] Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13(6), 377–387 (Jun 1970)

[10] Dadzie, A.S., Rowe, M.: Approaches to visualising linked data: A survey. Semantic Web 2(2), 89–124 (2011)

[11] Ell, B., Harth, A., Simperl, E.: Sparql query verbalization for explaining semantic search engine queries. In: Proceedings of the 11th Extended Semantic Web Conference (ESWC '14). pp. 426–441. No. 8465 in LNCS, Springer, Heidelberg (2014)

[12] Fu, B., Noy, N.F., Storey, M.A.: Indented tree or graph? a usability study of ontology visualization techniques in the context of class mapping evaluation. In: The Semantic Web–ISWC 2013, pp. 117–134. Springer (2013)

[13] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Pearson Education (1994)

[14] Hart, G., Dolbear, C.: Linked Data: A Geographic Perspective. CRC Press, Boca Raton (2013)

[15] Janowicz, K., Scheider, S., Pehle, T., Hart, G.: Geospatial semantics and linked spatiotemporal data - past, present, and future. Semantic Web 3(4), 321–332 (2012)

[16] Jones, J., Kuhn, W., Keßler, C., Scheider, S.: Making the web of data available via web feature services. In: Huerta, J., Schade, S., Granell, C. (eds.) Connecting a Digital Europe Through Location and Place, pp. 341–361. Lecture Notes in Geoinformation and Cartography, Springer International Publishing (2014)

[17] Kadlag, A., Wanjari, A., Freire, J., Haritsa, J.: Supporting exploratory queries in databases. In: Lee, Y., Li, J., Whang, K.Y., Lee, D. (eds.) Database Systems for Advanced Applications, Lecture Notes in Computer Science, vol. 2973, pp. 594–605. Springer Berlin Heidelberg (2004)

[18] Kauppinen, T., de Espindola, G., Jones, J., Sanchez, A., Gräler, B., Bartoschek, T.: Linked Brazilian Amazon Rainforest Data. Semantic Web Journal 5(2) (2014)

[19] Keßler, C., Janowicz, K., Kauppinen, T.: spatial@linkedscience - exploring the research field of giscience with linked data. In: Xiao, N., Kwan, M.P., Goodchild, M., Shekhar, S. (eds.) Geographic Information Science, Lecture Notes in Computer Science, vol. 7478, pp. 102–115. Springer Berlin Heidelberg (2012)

[20] Kovarsky, J.: Searching for early maps: Use of online library catalogs. ACMLA Bulletin (138) (2011)

[21] Kuhn, W., Kauppinen, T., Janowicz, K.: Linked data - A paradigm shift for geographic information science. In: Geographic Information Science - 8th International Conference, GIScience 2014, Vienna, Austria, September 24-26, 2014. Proceedings. pp. 173–186 (2014)

[22] Kveladze, I., Kraak, M.J., van Elzakker, C.P.: A methodological framework for researching the usability of the space-time cube. The Cartographic Journal 50(3), 201–210 (2013)

[23] Mandel, T.: The Elements of User Interface Design. John Wiley & Sons, Inc., New York, NY, USA (1997)

[24] Marchionini, G.: Exploratory search: From finding to understanding. Commun. ACM 49(4), 41–46 (Apr 2006), http://doi.acm.org/10.1145/1121949.1121979

[25] Möller, K., Dragan, L., Ambrus, O., Handschuh, S.: A Visual interface for building SPARQL queries in Konduit. In: Bizer, C., Joshi, A. (eds.) Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC-2008). CEUR-WS.org, Karlsruhe, Germany (2008)

[26] Russell, A., Smart, P.R.: NITELIGHT: A graphical editor for SPARQL queries. In: Bizer, C., Joshi, A. (eds.) Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany, October 28, 2008 (2008)

[27] Scheider, S., Degbelo, A., Kuhn, W., Przibytzin, H.: Content and context description - how linked spatio-temporal data enables novel information services for libraries. gis.Science (2014), forthcoming

[28] Scheider, S., Jones, J., Sánchez, A., Keßler, C.: Encoding and querying historic map content. In: Huerta, J., Schade, S., Granell, C. (eds.) Connecting a Digital Europe Through Location and Place, pp. 251–273. Lecture Notes in Geoinformation and Cartography, Springer International Publishing (2014)

[29] Schraefel, M., Karger, D.: The pathetic fallacy of rdf. In: International workshop on the semantic web and user interaction (SWUI). vol. 2006 (2006)

[30] Sequeda, J.: Introduction to: Triplestores [http://semanticweb.com/introduction-to-triplestores_b34996; last accessed: December 03, 2014] (2013)

[31] Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: Proceedings of the 1996 IEEE Symposium on Visual Languages. pp. 336–343 (1996)

[32] Smart, P.R., Russell, A., Braines, D., Kalfoglou, Y., Bao, J., Shadbolt, N.R.: A visual approach to semantic query design using a web-based graphical query designer. In: Gangemi, A., Euzenat, J. (eds.) Knowledge Engineering: Practice and Patterns (EKAW 2008). pp. 275–291. Springer-Verlag Berlin Heidelberg, Acitrezza, Italy (2008)

[33] Usbeck, R.: Combining linked data and statistical information retrieval. In: The Semantic Web: Trends and Challenges, pp. 845–854. Springer (2014)

[34] White, R.W., Roth, R.A.: Exploratory Search: Beyond the Query-Response Paradigm. Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan & Claypool Publishers (2009)

[35] Zviedris, M., Barzdins, G.: Viziquer: A tool to explore and query SPARQL endpoints. In: The Semanic Web: Research and Applications - 8th Extended Semantic Web Conference, ESWC 2011, Heraklion, Crete, Greece, May 29 - June 2, 2011, Proceedings, Part II. pp. 441–445 (2011)