Towards the Design of Expressive Data Exploration Environments

Thiago Nunes and Daniel Schwabe

Department of Informatics
Pontifical Catholic University of Rio de Janeiro
R.M.S. Vicente 225
Gávea Rio de Janeiro, RJ, Brazil
+55 21 3527-1500
{tnunes, dschwabe}@inf.puc-rio.br

Abstract. Information exploration processes are usually recognized by their inherent complexity, lack of knowledge and uncertainty, concerning both the domain and the solution strategies. Even though there has been much work on the development of computational systems supporting exploration tasks, the lack of a formal understanding of the exploration process and the absence of a proper separation of concerns approach in the design phase is the cause of many expressivity issues and serious limitations. This work shows how the design space for exploration environments can be structured by applying the separation of concerns approach, with special emphasis in characterizing the issues that must be addressed when designing the interface of such environments.

1 Introduction

Information exploration environments aim at supporting information gathering tasks that often involve a high degree of complexity, the lack of user's knowledge about the data, which is spread over multiple items and data types, and do not have a clear ending [20]. Exploration tasks usually involve sequences of data interactions that eventually lead to the desired outcome, which can be either a set of items or a knowledge state acquired along the process [5]. These particular characteristics differ exploration tasks from usual information retrieval tasks, where the user is assumed to know precisely how to translate his/her information needs into a query and the interaction is restricted to isolated sequences of query-responses [4]. As an example, the task "write a paper about recently discovered treatments for diabetes" would require exploratory behavior, while the task "discover who invented the light bulb" can be solved in a single query against search engines. Given the complexity of the tasks, exploration environments should be carefully designed to support a rich enough variety of data processing actions and strategies, accessed through their interfaces.

A widely accepted motivation for exploration behavior is usually the need for filling knowledge gaps that prevent the user to achieving his/her goals. Explorers tends to experiment high degrees of uncertainty along the process, mostly concerning whether the desired knowledge state can be achieved given the available information space, task constraints, and the computational support for data manipulation. Despite

the massive publication of semi-structured information, leveraged by the Linked Data community[1], and the amount of exploration environments available, it is still hard to assess to which extent an environment fits the exploration needs of the users.

In order to address this central research question, in previous works we proposed the adoption of the separation of concerns principle, guided by Normans' gulf traversal theory [12], as a means to improve the discussions of the concerns involved in the design of exploration tools. In that work we traced a parallel between Norman's semantic and articulatory distances, and the distinction between the exploration operations and their realizations in the interface, which usually was not considered or mentioned in state-of-the-art tools addressing exploration tasks [14]. Subsequently, we proposed a first approximation of a framework of exploration operations expressive enough to describe the majority of state-of-the-art tools, which allowed us to analyze and compare the functional aspect of exploration tools, abstracting interaction and interface issues [16]. As an example, consider a refinement operation over a set of information items. A functional concern for refinements is whether the tool allows disjunctions, conjunctions, or negations of filters. On the other hand, an interaction/interface concern is which interface controls and interaction dialogues support the specification of filters and logical connectors.

The proposed framework presents a rich semantics for describing the functional aspect of exploration tools, and may guide the design of interaction dialogues and interface widgets. Nevertheless, a remaining open question is, how to approach the interaction/interface design space based on a formal layer of exploration operations?

This work presents a novel way to approach the design space for interfaces of exploration environments, by leveraging an expressive formalization of exploration actions and processes. To illustrate the discussions of the design issues and possible solution alternatives, we use the case of the design and implementation of the XPlain environment, which provides higher expressivity when compared to state-of-the-art tools, as we demonstrate in [13]. The example uses the Linked Data OpenCitations dataset[1].

The contribution of this work is a framework that presents a novel approach to characterize the interaction/interface issues. This framework serves as a guideline for designing expressive exploration environments over semi-structured data. Even though we discuss the issues and possible solutions in terms of a real exploration environment, the framework allows abstractions that can be generalized to any exploration environment design.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 presents the separation of concerns approach applied to exploration environments, leading to a three-layered architecture. Section 4 presents the data model used in the definition of the exploration operations. Section 5 addresses issues related to exploration actions and strategies. Section 6 addresses interaction/interface design issues. Finally, we present the conclusions and future steps.

[1] http://linkeddata.org/

2 Related Work

The need of separating visual representations from processing operations has been established in the visualization area presenting taxonomies, typologies, and ontologies addressing at least these two concerns [2, 3, 7, 10, 18]. Chi's work [10] divides the design of a visualization system in a sequence of stages in a pipeline that receives raw data as input and generates interactive visualization of the raw data as output. Each pipeline stage receives a set of data items, applies processing operations to transform the data, and passes the transformed data to the next stage. "scroll", "zoom", "filter", "rotate", and "scale" are operations that can be carried out in the view resulting from the pipeline. The work in [2] presents a taxonomy of analytic operations for describing visualization tasks containing, for example, "Cluster", "Filter", "Sort", and "Correlate" operations. The work in [7] presents a typology of abstract visualization tasks addressing the *Why*, *How*, and *What* aspects of a visualization task independently of the kind of visualization and of the task domain. The *Why* concerns the goals, such as "discover" new information, "present" data, or "explore". The *How* presents the actions to achieve the goals, such as "select", "navigate", "filter", and "aggregate". The *What* describes input and output resources handled by the tasks. These approaches are valuable to promote some degree of separation between the description of users goals, tasks, and operations from visual encoding details but their lack of formality makes it hard to analyze where they overlap and what are the differences. Moreover, they do not present detailed discussions of interaction/interface design issues with respect to a given well-defined conceptualization of exploration processes and strategies.

Visualization systems are concerned with encoding data in a visual representation to foster human cognitive perception. Although interactive visualizations can be used to explore a dataset to some extent, supporting exploration behavior is not its main goal [19]. Moreover, even in interactive visualizations, the user is usually restricted to a specific visual representation of the data aiming at highlighting a certain set of dimensions. Since exploratory search tasks tend to be general and multifaceted [20], it is very difficult to know in advance which data dimensions will suffice. Therefore, an exploration environment should support a broader class of tasks that may even include sense-making activities and manipulations of the raw data in order to select proper dimensions to encode in visual representations. In this context, one advantage of our reference framework is that it allows designers to situate visualization concerns in the design of exploration environments. Within the framework, the projection of the data onto a visual counterpart along with interaction controls and dialogue structures is a concern of the interaction/interface layer. The data processing interactions can be designed on top of the same framework of operations whose inputs are visually encoded items and relations.

Beyond the works in visualization field, there are some works addressing issues related to the broader exploration field. The work in [6] presents a similar architectural view of exploration environments and also abstracts the functional aspects in the SeCoQL exploration language. However, it presents a restricted set of operators containing only refine, pivot, and ranking, and does not approach interface

concerns in detail. The works in [1] and [21] propose a separation of interface details from the underlying features, but there the main goal is to present a common evaluation framework for comparison purposes and not a detailed separation of concerns discussion. Moreover, the features are not formally defined, which is the source of many ambiguities, such as different interactions for an operation being mistakenly understood as being distinct features.

3 Reference Framework

A serious problem of exploration tools is the lack of a clear separation of the exploration operators from their realization in the interface [15].

We propose the organization of the design space of exploration tools in a three-layer architecture: Data Access, Functional, and Interaction/Interface. The Data Access layer is responsible for providing access to data repositories and mapping their data models to an abstract data model, which will be manipulated by the user throughout the exploration task. The Functional layer presents a set of exploration operators whose repeated compositions capture the solution strategies adopted by the users. The Interaction/Interface layer provides proper access to both the operators and the data items being manipulated. Fig. 1 shows a conceptual view of the design space.

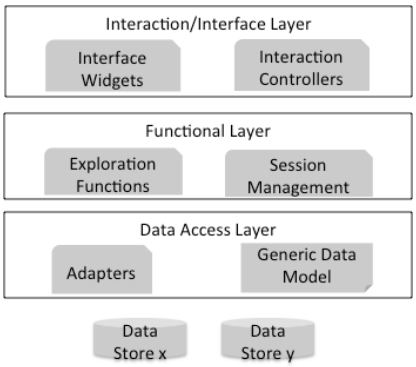

Fig. 1. The layers of an exploration environment architecture.

The next sessions discuss the design questions for each layer of the architecture. We discuss solution ideas for each question in terms of the design and implementation of a generic and expressive exploration environment XPlain.

4 Data Model

The design questions addressed by this layer are, how to provide access to the data repositories and how to abstract the details of the data model adopted by the repositories so that the same exploration actions can be executed over data represented in different ways.

Independently of how the data is represented, the user effectively manipulates items and relationships among them. Items can be organized in groups, such as papers by author or papers by venue. Groups can also be formed along more than one dimension, such as papers by author by publication year. Since such structures are not directly mapped to traditional data models we chose to model items and relations as nested relations. As an example, "papers by author by publication year" is a three-level nesting, having the papers grouped by year inside a group for each author. Nesting relations can be represented as trees, as Fig. 2 shows. We call "exploration set" any nested relation generated by the execution of an exploration action.

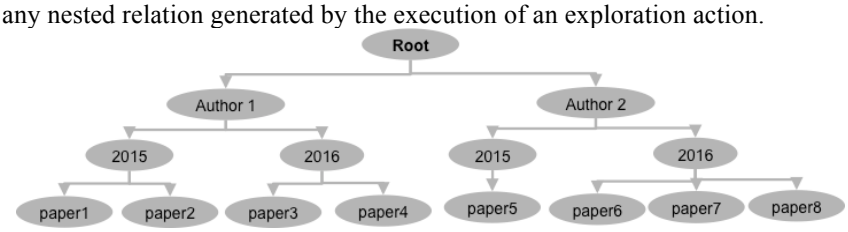

Fig. 2. Nesting of papers by author by publication year.

This data model is very similar to the one used in NoSQL document-oriented databases [9], where the nestings of an exploration set can be mapped to collections and nested arrays. It also can be translated to the RDF model, where each path from the root to the leaves can be mapped to a set of triples, using the reification mechanism to describe statements over statements in case of trees having more than three levels.

5 Functional Layer

The main concern of this layer is to define exploration actions as operations over the generic data model presented in section 4 and the exploration process as a composition of these operations. In order to find an expressive set of exploration actions we carried out detailed analyses of a significant, if not the majority, of the state-of-the-art tools published in the literature. A summary can be found in [16]; for a detailed description of operations design process, refer to [13].

The operations of the functional layer are defined in terms of items and relations. A dataset $D = <I, R>$ is defined as a set of items $I = \{i_1, i_2, ..., i_n\}$ and a set of pairs of items $R = \{<i_1, i_2>, ..., <i_j, i_k>\}$, representing the relationships in the dataset.

We briefly present each proposed operation in terms of its input parameters, result sets, and examples of use over the dataset defined before. The complete formalization of the operations is beyond the scope of this paper and can be found in [13]. The operations' descriptions are as follows:

- *Pivot(S, Rel):* maps the leaf items of *S* onto a set of related items through the relation *Rel*;
- *Refine(S, Filter):* restricts the leaves of *S* keeping only the items that match the predicate *Filter*;

- *Group(S, grel):* maps each leaf item of *S* onto their group keys using the grouping relation *grel*, where, the leaf items are nested within their respective group keys;
- *Correlate(S, T):* finds all paths between every source item in *S* to every target item in *T* (many-to-many). A path is a set of items that connects a source to a target item and, each path is a different nesting in the result set;
- *Rank(S, level, score):* ranks a given nested level of *S*, where the relevance of the items are obtained by a score function;
- *Map(S, mRel):* maps the items in *S* onto another set of items using the computed relation *mRel, where, mRel* is a relation/function provided by the environment, such as counts and format and scale converters;
- *Unite(S, T):* receives two exploration sets and applies a set union to each level of *S* and *T*;
- *Intersect(S, T):* receives two exploration sets and returns a tree that is the intersection between the levels of the input set trees*;*
- *Diff(S, T):* receives two exploration sets and applies a set difference between the leaves of *S* and the leaves of *T*.

We call auxiliary functions or auxiliary relations all functions/relations that are not defined as an exploration operator, such as, mapping relations for *Map*, score functions for *Rank*, and filtering predicates for *Refine*.

In summary, independently of the interface design, the range of solution strategies for exploration tasks is directly related to the set of operations supported and the possibilities of combinations of these operations. This is the main concern of this layer. For details and more examples of applications of the operations in a real case, refer to the case study published in [15].

6 Interaction/Interface

Having defined the operators and their possible combinations for exploration tasks, the interface designer must focus exclusively on deciding which interaction paradigms are more adequate. This section presents a discussion of interaction/interface issues separating its concerns from the exploration actions and compositions defined by the functional layer. Based on the concepts of the functional layer, the main goals of the interface are: (1) to present one or many exploration sets, where the items of each set may be hierarchically organized (nested); (2) Allow the user to select an operator/composition and specify its parameters; (3) Allow the user to visualize, manage, and browse the exploration trail. In order to leverage the discussion, we used a real problem situation discussed in [11] in the scientific publications field using the Open Citations [17] Linked Data dataset. Fig. 3 shows a screenshot of the interface of the XPlain environment. A screencast of a session can be seen at https://vimeo.com/227356693.

Next, we discuss the interaction/interface design issues in the light of the separation of concerns approach, using the XPlain interface to exemplify the issues and possible solution ideas.

6.1 Requirement 1: Manipulation of Exploration Sets and Items

The first challenge for the design of exploration environment interfaces is how to present the data being manipulated and its relationships. The biggest issue to be dealt with is handling the potential excess of information to be presented, as the number of items can be very large. Here, Shneiderman's visual information seeking mantra "Overview first, zoom and filter, then details-on-demand" [18] should be considered as a guideline.

Considering the conceptualization of the exploration process as a functional composition that results in multiple exploration sets, the design alternatives for presenting those sets are: show one exploration set at a time (unifocal) or show many sets at a time (multifocal).

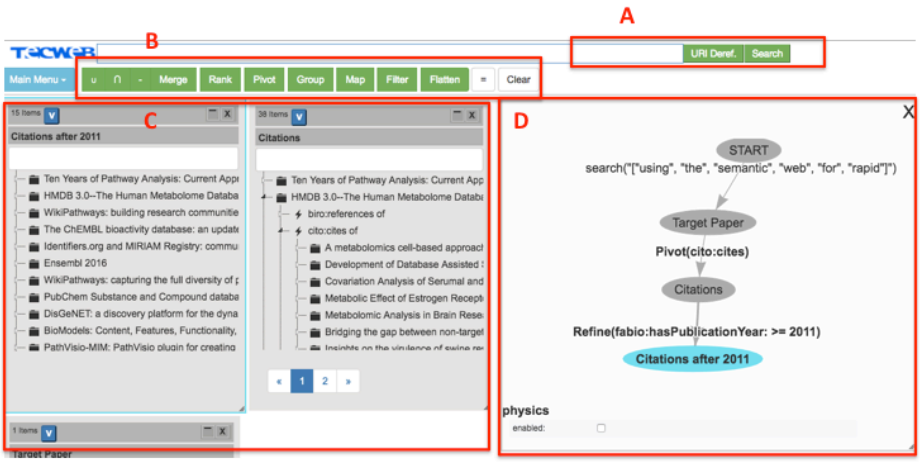

Fig. 3. The interface of the XPlain environment. (A) keyword search controls; (B) Exploration operations toolbar; (C) Exploration sets area; (E) Exploration trail view.

Unifocal interfaces have the advantage of reducing the amount of information shown at a given time and requiring less focus management interactions. However, they do not properly support operations that take more than one set as input, such as comparisons of alternatives [8]. For example, imagine a user interested in comparing the publication profiles (e.g., venues in common) of two researchers in a certain period of time. S/he filters the publications of each researcher by the desired period, pivots by venue, and computes the intersection (or difference) of the two sets. In a unifocal interface s/he must apply this sequence of operations to each researcher one at a time, and somehow apply the intersection operation (if available in the functionality layer) to the results, which won't be both available in the same interface. Note that if the functionality layer does not provide a set operation, s/he must annotate the results and then make the comparisons offline. For web browsers, a common strategy is to open two or more windows and organize the windows to support the comparisons, but this is limited by available screen real estate.

In a multifocal view s/he is able to visualize the two sets of venues simultaneously and compute the intersection straightforwardly. The drawback of multifocal interfaces is the need to design of focus management controls, such as maximization, minimization, and restoration controls to avoid information overload. If comparisons between alternatives are not the case or the device is very restricted in screen size, unifocal interfaces may be more appropriate.

An additional design issue is the layout and presentation of the relationships between the sets on the screen. To illustrate one possible set of options, in XPlain we opted for a multifocal interface to better support operations over multiple sets, where each set presentation has minimization/maximization controls. Fig. 3C shows two exploration sets in the workspace. The last generated set is always placed on top of the screen and the exploration trail presents the relationships between the sets and also allows the user to navigate to intermediary sets by clicking on the corresponding node in the graph. After deciding between unifocal and multifocal presentations, it is necessary to define the organization and interactions for the exploration items and the relations they participate in, within each exploration set.

According to the functional model, there are two types of item relations that must be considered: schema relations and computed relations. Computed relations are relations created along the exploration process, such as grouping relations or mappings, which do not necessarily have an identifier, such as predicate URIs in RDF or column names. Two common representations of schema relations found in the literature are tabular format, where each relation becomes a column, and graph format, where the relations are the edges between nodes. A tabular format may be easier for spreadsheet users while graph views favors the visualization of the joins between the items.

In XPlain we took a different approach by adopting a directory metaphor, where items are mapped to directories and both schema and computed relations are nested directories, as shown in Fig. 4. This choice allows a natural representation of groups, where each group is represented as a separate directory. The drawback is the visualization of items that participate in more than one relation, as items related to two different nested items will appear repeatedly, in two "directories".

Even in a unifocal interface, the number of items within a single exploration set can be large, so the designer should weight alternative choices for presenting the set using scroll or pagination controls.

It is also typically desirable to apply some natural ordering to the items. Although our model describes ranking as an independent exploration operator, it can also be used in conjunction with other operators. Thus, even when operators other than *ranking* are selected, such as *keyword refine* or *grouping*, the interface can also make the composition with a ranking function and send the result to the server in order to enforce a natural ranking for the result set. In XPlain we opted for pagination controls with a limit of twenty items per page and an alphabetical or numerical ordering of results.

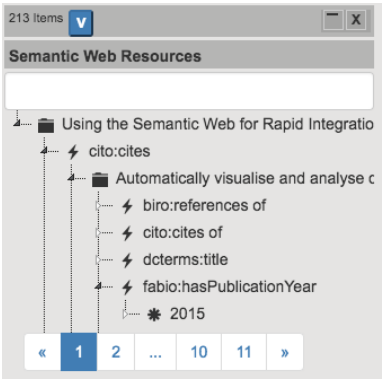

Fig. 4. Visual representation of an exploration set as a nesting of items and relations.

In summary, the interface design issues for the manipulation of exploration sets and items are:

1. Choosing between unifocal and multifocal view
   a. If multifocal: design appropriate focus management controls for the sets, such as maximization, minimization, restore, and hiding controls;
2. Deciding where and how to show relationships between exploration sets;
3. Designing focus management controls for the items within the set e.g. pagination, scroll, or a combination of both;
4. Determining the best visualization for items and items relations (schema and computed): graphs, trees, tables, lists, etc;
5. Establishing a natural order for presenting exploration items and, possibly, adding sorting controls.

6.2 Requirement 2: Applying Exploration Operations

The application of exploration operations presents another class of interface/interaction design issues, which concern both the selection and activation of an operator, and the definition of its parameters. The functional layer defines four types of arguments: exploration items, auxiliary functions, relations, and relation paths. Next, we argue that each argument type may require distinct interaction models.

To invoke an exploration action the user must assign the values to each input parameter of the invoked operation. Each assignment is a binding, i.e., a pair <Parameter, Value> that will be evaluated when operation is executed. For example, pivoting requires two bindings: the definition of the input set and the pivoting relation. For binding definitions, the interaction issues are: defining the assignment order for parameters, and defining the interaction that will support the binding definition. The latter issue depends on both the argument type and the operation.

With regards to the order of the assignments, consider the pivoting action as an example. Some design alternatives are: the user selects the input set, activates the pivoting operator, and the system shows the relations for selection (e.g., interaction in

SeCo[6]); Alternatively, when the user selects the set, the interface could show all relations as selectable elements whose activation causes a pivoting over the selected relation. For tabular presentations, the first alternative may be better due to layout organization issues, however, for graph and list presentations the second option seems to be closer to hypertext browsing, which may favor Web users. The second option is the solution adopted by the majority of faceted search tools with pivoting functionality.

Another example is the definition of bindings for the *Refine* operation, where the user should select the filtering relation, the filtering predicate and a value. One option is to simply allow the selection of values, where the relation is inferred and the filtering predicate is always an equality test. Another option is to allow the selection of the relation, the value, and the filtering predicate, which may be different than equality comparison e.g., greater than or less than operators. Thus, there can be many distinct interaction sequences for the definition of the bindings for each operation.

The next issue concerns specific interactions for different types of parameters. Considering the case of the *Refine* operation where the user must define bindings for the relation or the relation path, the comparison operator (e.g. =, <, >), and the restriction value for the relation. With regards to the relation, the interface has to allow both the selection of relations and of relation paths.

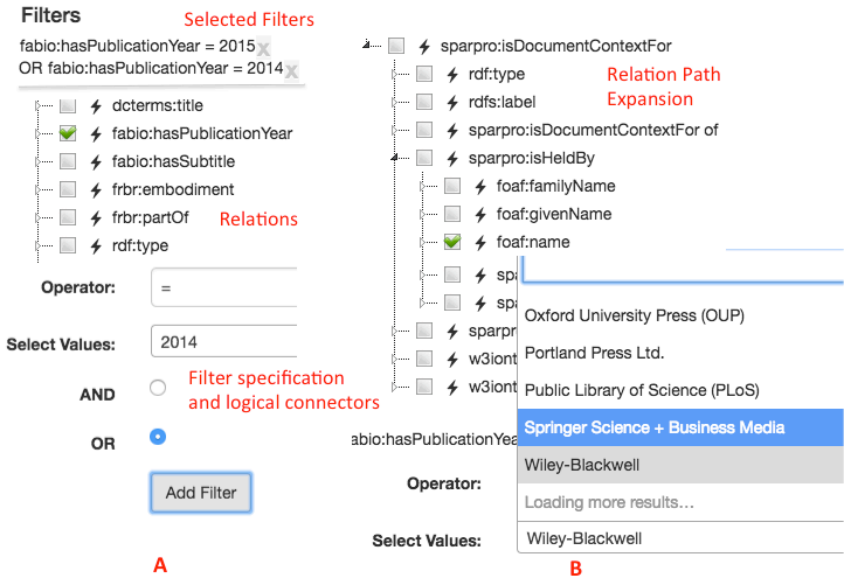

Fig. 5. View for the *Refine* operation. The user selects relations (A) or relation paths (B) and restriction values for each filter. Filters can be disjunctive or conjunctive according to the selected logical operator.

For example, in Open Citations, if we want to refine papers by venue names, we must bind the relation path *:isDocumentContextFor:isHeldBy:name* to the relation

parameter, as shown in Fig. 5B. One design option is to allow the user to pivot relation by relation in this path until reaching the next to last relation, which is the *:isHeldBy* relation in this case. At this point, the interface can show the possible relations and values for the holders, which includes the *:name* relation and the actual venue names for selection. The computation is, therefore, carried out relation by relation until the desired path is achieved. Next, a selection of a venue name will cause the refinement over the path *:isDocumentContextFor:isHeldBy:name*. This is the most common interaction for path refinements found in faceted search interfaces, but considering the size of paths, the possibility of mistakes, and the amount of refinements required for the task, this design can be cumbersome. Another option is to allow the visualization of relation chains on demand, where the user can explore and select relation paths without causing a context change (pivoting). XPlain implements this design option with relation nestings built at runtime, where when a relation *:x* is nested with a relation *:y* then there is relation path *:x:y* in the dataset representation. This design allows the user to browse the nestings in order to find the desired path, with reasonable performance. Fig. 5B shows the nesting of *:isDocumentContextFor*, *:isHeldBy*, and *:name* relations.

Auxiliary functions, such as the comparison operators, the scoring function, and the mapping functions can be picked out in the interface from a pre-defined set. For example, the *Refine* modal dialogue presents a selection box with all comparison operators available, as Fig. 5A shows. However, since it is very difficult to define a complete range of functions that covers all problem domains, the interface can also allow the user to describe the function in some computable language. Consider a user wishing to convert a set of measures to a different scale. The functional layer provides the *Map* function for such tasks, but, the desired scale converter is not among the available mapping functions. The user could simply type the formula and the interface creates the binding. Therefore, the interaction design should not only consider interface selection, but also textual inputs, with some validation in the case of function definitions, and filterable selection lists. The same issue also occurs for bindings of exploration items. XPlain's interface allows the definition of new auxiliary functions using a Domain Specific Language (DSL) implemented by the functional layer. We also chose filterable selection lists for the definition of the filtering values, as shown in Fig. 5B.

So far we have discussed the interaction/interface design issues and possible solution ideas for the execution of single operations. For some recurring functional compositions, there can be alternative interaction styles to the operation-at-a-time approach. An example of such compositions can be found in the expansion of an exploration item, shown in Fig. 4. When a user double clicks an item in the exploration set, XPlain executes a composition of *Refine* and *Pivot* to respectively select the clicked item, and pivot to its set of relations. The relations are shown as nested items that can also be expanded. This interaction allows the user to browse the graph of relations of an item in a follow-your-nose style without causing a context change or the addition of a new exploration set for each *Refine* and *Pivot* executed. This illustrates the fact that the designer should explore alternative interaction and

interface designs for combinations of operators that are more appropriate for a given task context.

Capturing common combinations that would require more appropriate interaction designs is likely to be difficult, since the combinations may not be obvious for some domains and contexts. However, we expect that these combination patterns will emerge with the continued use of the environment. Since the patterns are formally described and recorded, they can be mined from the environment log and analyzed from the perspective of separate interface and interaction models.

The interaction/interface design issues presented for the specification of exploration operations are:

1. The ordering for the specification of bindings for each operation.
2. The interaction required for specifying bindings for each parameter, depending on the parameter type and the operation. Some possibilities discussed were: interface selection, textual inputs, filterable selects, computable specification for auxiliary functions, and navigation through relation paths for relation parameters.
3. The possibility of modeling particular interactions for specific combinations of operators, such as for combinations of *Refine* and *Pivot*, and for compositions of *Refine, Intersect, and Unite* that can be modeled as a faceted search interaction.

6.3 Requirement 3: Exploration Trail Management and Browsing

It has been recognized that exploration tools should allow the user to visualize the history of the exploration actions [19]. The functional layer defines relationships between result sets, where the result set of a previous action can serve as the input for the next. Hence, the design issues at this point are how to present the exploration trail and how to allow its manipulation.

Some interface options for visualizing the exploration trail are trees or graphs. The tree representation has the advantage of allowing the user to collapse or expand the branches, which may be a good option considering the "details-on-demand" rule of the information seeking mantra. However, since the functional layer presents operations that receive two sets as input (e.g., unite, intersect, and diff), tree representations present a drawback because the result set of these operations must be repeated in two branches. With tree representation is not easy to perceive these join nodes, i.e., sets resulting from combinations of two input sets. In XPlain we choose a graph representations in order to enhance the perception of join nodes. For the following examples, consider the case of a user reviewing a paper. One revision strategy is to find relevant papers of the same area of the reviewing paper that were not referenced. Fig. 6 shows an exploration trail example for the case study of "finding relevant and not cited papers". The join node is the set difference operation.
The graph in Fig. 6 is a visual representation of the sequence of operations applied along the exploration process, where, each node is a result set and the arrows represent the operations applied. The "START" node represents the whole dataset and the highlighted node "Relevant and not Cited Papers" is the result of the difference

between the citations is the result of the difference between the citations of the paper being reviewed and the top 20 most relevant papers of the Semantic Web area, according to the number of income citations.

The graph in Fig. 6 is more than just a visual representation of the exploration trail - it can also be used as a first-class object, where the user can parameterize the operations and reevaluate dependent branches. For example, the user could replace the set "Semantic Web Papers" in the exploration trail in Fig. 6 by a set of papers in another research field and reevaluate the entire branch, thus reusing an exploration trail for different papers of distinct research fields. In other words, it is possible to reapply strategy used to solve a task as represented by the exploration trail.

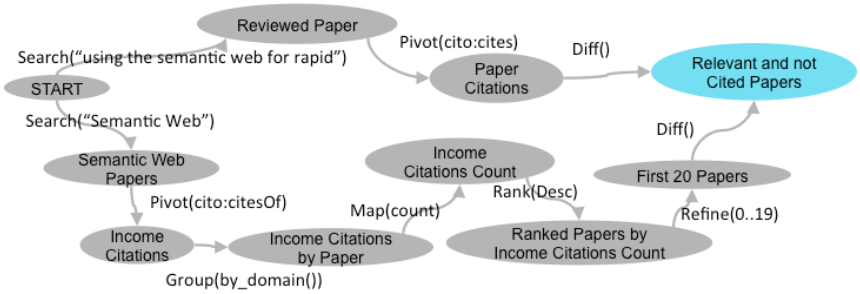

Fig. 6. Graph representation of the functional composition for the task "finding relevant and not cited papers".

Once we recognize that an exploration is, in the end, also a function, the interaction issues for allowing the reevaluation of a functional composition become quite similar to the issues concerning the definition of bindings for the operations presented in the previous section. The additional step is to consider the union of bindings from all operations in the composition as bindings of the exploration. Therefore, the reevaluation of the functional composition requires the redefinition of one or many bindings of some operations. The interface could show the bindings and ask which ones must be replaced for the reevaluation. The same design decisions adopted for the definition of bindings for each argument type also apply for the redefinition of bindings of functional compositions[2].

The interaction/interface design issues for exploration trail management and browsing are:

1. The visual encoding for the exploration trail, which includes both exploration sets and their dependencies;
2. Allow the user to browse the exploration sets in the nodes of the exploration trail;
3. Allow the user to access the bindings for specific operation/functional compositions;

[2] This feature is currently under development

4. Define the interaction for binding redefinitions and reevaluations from the exploration trail.

In summary, we have shown how separating the concerns of interaction/interface design from the operations of the functional layer, together with use of the functional layer as a guide for what the interface should provide for specific task contexts guides the discussions of interaction possibilities. Since the main concern of this work is to explore the design space of exploration environments, the XPlain interface is one possible interface and interaction model for the functional layer that, even though it presents full expressivity, it may not be efficient for all exploration contexts and users.

7 Conclusion and Future Directions

This work presents a novel way to approach the design space of exploration environment interfaces based on the separation of interface/interaction aspects from the exploration operations and compositions, and data access concerns. The main goal is to present and discuss abstract design issues and solution ideas for the interaction design of exploration environments from the perspective of a well-defined and expressive framework of exploration actions. We also based the discussion on the design and development of a new expressive exploration environment in order to demonstrate the occurrence of the design issues and possible solutions in a realistic design case.

As future directions we plan to execute user studies and analyze the usage of each operator within an exploration case. Given the agreement on the operators and compositions required, usability studies must be carried out for devising efficient interfaces and interaction dialogs.

Moreover, we plan to investigate the benefits of the reuse of functional compositions in future explorations.

Acknowledgement

The authors were partially supported by CNPq project 557128/200-9 National Science, Technology Institute on Web Science, CAPES, and Google Research Program.

8 References

1. Alahmari, F. et al.: Evaluating Semantic Browsers for Consuming Linked Data. Proceedings of the Twenty-Third Australasian Database Conference - Volume 124. pp. 89–98 Australian Computer Society, Inc., Darlinghurst, Australia (2012).
2. Amar, R. et al.: Low-level components of analytic activity in information visualization. Proc. - IEEE Symp. Inf. Vis. INFO VIS. 111–117 (2005).
3. Amar, R. a., Stasko, J.T.: Knowledge precepts for design and evaluation of information visualizations. IEEE Trans. Vis. Comput. Graph. 11, 4, 432–442 (2005).

4. Bates, M.J.: The design of browsing and berrypicking techniques for the online search interface. Online Inf. Rev. 13, 5, 407–424 (1989).
5. Belkin, N.J. et al.: Ask for Information Retrieval: Part I. Background and Theory. J. Doc. 38, 2, 61–71 (1982).
6. Bozzon, A. et al.: Exploratory search framework for Web data sources. VLDB J. 22, 5, 641–663 (2013).
7. Brehmer, M., Munzner, T.: A multi-level typology of abstract visualization tasks. IEEE Trans. Vis. Comput. Graph. 19, 12, 2376–2385 (2013).
8. Buschbeck, S. et al.: Parallel faceted browsing. CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13. p. 3023 ACM Press, New York, New York, USA (2013).
9. Cattell, R.: Scalable SQL and NoSQL data stores. ACM SIGMOD Rec. 39, 4, 12 (2011).
10. Chi, E.H.: A Taxonomy of Visualization Techniques using the Data State Reference Model. Inf. Vis. 2000. InfoVis 2000. IEEE Symp. 94301, Table 2, 69–75 (2000).
11. Di Iorio, A. et al.: Exploring Bibliographies for Research-related Tasks. Proceedings of the 24th International Conference on World Wide Web - WWW '15 Companion. pp. 1001–1006 ACM Press, New York, New York, USA (2015)
12. Norman, D.A., Draper, S.W.: User Centered System Design; New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc., Hillsdale, NJ, USA (1986).
13. Nunes, T.: A Model for Exploration of Semi-Structured Datasets. Pontifical Catholic University of Rio de Janeiro (2017).
14. Nunes, T., Schwabe, D.: Exploration of Semi-Structured Data Sources. 3rd Int. Work. Intell. Explor. Semant. Data (IESD 2014). (2014).
15. Nunes, T., Schwabe, D.: Frameworks for Information Exploration – A Case Study. 4th International Workshop on Intelligent Exploration of Semantic Data (IESD 2015). (2015).
16. Nunes, T., Schwabe, D.: Frameworks of Information Exploration - Towards the Evaluation of Exploration Systems. Proceedings of the 5th International Workshop on Intelligent Exploration of Semantic Data - IESD '16. , Kobe, Japan (2016).
17. Peroni, S. et al.: Setting our bibliographic references free: towards open citation data. J. Doc. 71, 2, 253–277 (2015).
18. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. Proceedings 1996 IEEE Symposium on Visual Languages. pp. 336–343 IEEE Comput. Soc. Press (1996).
19. White, R.W., Roth, R.A.: Exploratory Search: Beyond the Query-Response Paradigm. Synth. Lect. Inf. Concepts, Retrieval, Serv. 1, 1, 1–98 (2009).
20. Wildemuth, B.M., Freund, L.: Assigning search tasks designed to elicit exploratory search behaviors. Proc. Symp. Human-Computer Interact. Inf. Retr. - HCIR '12. C, 1–10 (2012).
21. Wilson, M.L. et al.: Evaluating advanced search interfaces using established information-seeking models. J. Am. Soc. Inf. Sci. Technol. 60, 7, 1407–1422 (2009).