3
A Framework of Reference

State-of-the-art environments in exploration area often present conceptual architectures, but details of the design process and methodology are not present (BOZZON, A *et al.*, 2013; BOZZON, ALESSANDRO *et al.*, 2010; GARCÍA *et al.*, 2013). In order to achieve a thorough understanding of the design space and build a reference framework to approach the design of exploration environments, we adopted a strict separation of concerns approach.

Separation of concerns is a widely recognized principle that allows designers to approach each aspect of a complex object in isolation, hence, abstracting them from the remaining concerns. Although the separation of user tasks, operations and goals from interface design details has been widely recognized as a valuable approach in HCI since the existence of task models, such as the Goals, Operations, Methods and Selection rules (GOMS) family (JOHN; KIERAS, 1996), it has probably not been applied in the context of exploration tools. One of the consequences is the difficulty to assess both the range of exploration tasks the tools are suitable for, and how well they support the user during an exploration task.

The separation of concerns we propose is consistent with Norman's theory of gulf traversal (NORMAN; DRAPER, 1986), which separates the user-system interaction in two major phases: the "execution gulf" and the "evaluation gulf". Figure 5 shows Norman's gulf traversals.

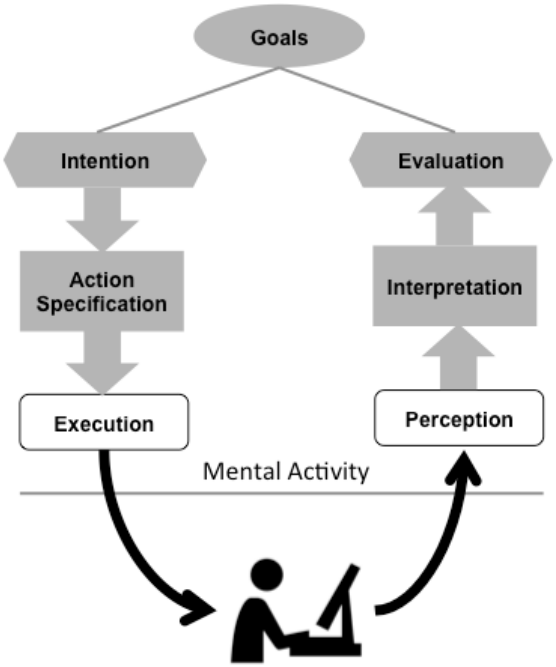

Figure 5 - Norman's execution and evaluation gulfs (NORMAN; DRAPER, 1986)

The "execution gulf" covers all the sequence starting with the user's intention of executing an action and ends with the translation of this intention in terms of interface controls. The "evaluation gulf" concerns the interpretation and assessment of the results generated by the "execution gulf". As an example of a real world gulf traversal, (NORMAN; DRAPER, 1986) presents the bathtub water flow and temperature control. The execution gulf starts with the goal of filling the bathtub with enough water in the desired temperature. There are two cognitive variables involved: water temperature and water flow rate, where both exist in the mind of the bathtub user. However, this task must be carried out in a physical system. Therefore, the user has to map the cognitive variables into the physical variables and their control mechanisms, e.g. faucets to control hot and cold water flows. In the evaluation gulf, the user has to perceive and interpret the function that relates the manipulation of the variables and the change in the physical state, e.g. the total flow is the sum of hot and cold flows and the temperature is the difference ratio between the hot and cold water flows. The cycle ends with the evaluation of whether the goal of filling the bathtub was achieved or not. If the user is not satisfied, s/he can carry out a new cycle of execution and evaluation gulf traversals

Gulf traversals can be considered bridging processes between mental variables, physical variables, and control mechanisms, governed by two distances: the semantic and articulatory distances. Figure 6 shows the concepts that must be bridged and the distances between them in a gulf traversal.

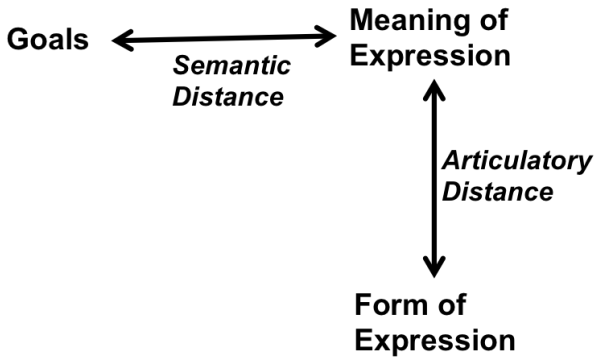

Figure 6 - Semantic and articulatory distances for gulf traversals (NORMAN; DRAPER, 1986)

While the semantic distance is the distance between the user's goals and intentions and the actual meaning of the physical variables, e.g. distance from water flow and temperature to hot and cold water flow, the articulatory distance stands between the physical variables meaning and the physical controls to execute them through the system interface, e.g. hot and cold water flow controlled by two faucets or a single faucet.

In the design of information exploration environments, we consider acquiring new knowledge by examining pieces of information as an ultimate goal (MARCHIONINI, 2006). Therefore, these are the cognitive variables that must be matched with physical datasets, containing items and relations, and data processing operators for exploring them, which we call exploration functions. The "Semantic Distance" is, therefore, the distance between the learning goal and the encoded information and operators available. The definitions of the exploration functions and the possible combinations relate to the "Meaning of Expression". The "Articulatory Distance" is the distance between the meaning of the functional compositions and their execution through interface controls and interaction dialogues, which are the "Form of Expression".

As an illustration of the semantic and articulatory distances applied to information exploration, consider the goal of filtering items that match a certain criteria. The filtering action and the range of criteria available through the

interface characterizes the “Meaning of Expression”. The user intentions and goals must be bridged with the range of filtering criteria available, which may not be possible if the semantics of the filtering language is insufficient. For example, imagine a user wishing to buy a new car and his/her intention is to select models of two distinct brands for comparisons. The user intention is to filter car models of either brand X or brand Y. However it is not rare to find e-commerce web sites that lack the possibility of applying disjunctive filters. If this is the case, the semantics of the interface language (Meaning of Expression) is insufficient to support this intention, which increases the semantic distance and turns the bridging of the “Goals” with the “Meaning of Expression” more difficult, if not impossible. Notice that, in this filtering case the user could follow some roundabout ways to accomplish this goal, such as opening two browser tabs/windows and applying the filters in different sessions, or even applying a filter to obtain models of brand X, use an external tool to save the results, and apply a new filter for brand Y. In both cases, additional complications are added due to the lack of expressiveness of the filtering expressions.

The articulatory distance is a measure of how difficult it is to articulate an expression using interface controls of the physical system. In the case of filtering car models of either brand X or brand Y, the user may specify the brands by typing in a text input or selecting it from an ordered list. The first option could increase the articulatory distance if the user does not know how to correctly spell the brand name. Another articulatory concern is how the user can specify the disjunction operator. Will it be the default filtering logical connector or the user would have to select the “OR” option in a checkbox, for example?

Considering the aforementioned design concerns and dimensions, this work proposes a design approach that aims at separating the issues related to the semantic distance from the issues related to the articulatory distance. In order to bridge the semantic distance, we devised a framework of exploration operations that formally defines the exploration process and the possible solution strategies. The articulatory distance can be approached through usability and communicability experiments (DE SOUZA, 1993), user-system interaction dialogue structures (BARBOSA; GRECO, 2003; BELKIN, N, 1995), and visualization techniques, which is beyond the scope of this research. Nonetheless,

we organize and discuss a list of general interface issues for exploration environments based on the concepts formalized by the exploration framework.

Using the separation of concerns principle, we propose to divide the design of exploration systems into three layers, as represented in Figure 7.

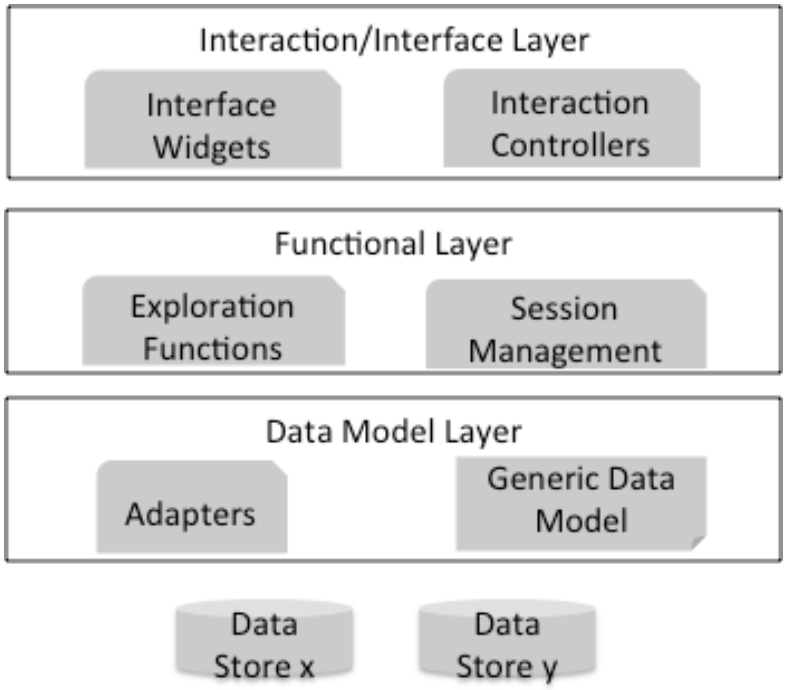

Figure 7 - The layers of an exploration environment architecture.

The Data Model layer addresses the design of data representation and access, which comprises a generic data model, storage and retrieval techniques. The second layer addresses the exploration functions, with their parameters and results, and the management of task progress, represented by the Session Management module. The Interaction/Interface layer addresses interaction and interface issues that support both the execution of the exploration functions and task management actions along the exploration process. The following sections discuss the responsibilities of each layer. We do not present the layers in the same order of Figure 7, though. Since the central research issue of this work is to define an expressive functional layer, we present its concepts first. Moreover, the functional layer establishes the basic semantics and requirements for the conceptualizations of both the Interface/Interaction layer and the Data Model layer.

3.1. Functional Layer

A common problem in the design and evaluation of exploration systems is how to assess the degree of expressivity of such tools. At the time of this research, there isn't a published common framework of operations that allows designers to identify neither which processing can be applied to a dataset nor what are the possible sequences. As a result, tools designed for exploration tasks are difficult to analyze and compare. For example, there are many cases of functionality overlapping, where the feature proposed has already been addressed with different interaction styles and interface signs (ARAUJO *et al.*, 2010; GARCÍA *et al.*, 2013; HUYNH; KARGER, 2009; POPOV *et al.*, 2011).

As an illustration, consider the pivoting action. Pivoting allows the user to change the current focus from one set of items to a related set of items, e.g. from publications to their authors. Pivoting can be found both in Tabulator (BERNERS-LEE *et al.*, 2006), Parallax (HUYNH; KARGER, 2009), and in SeCo (BOZZON, A *et al.*, 2013) tools, among others. Parallax and SeCo offer the same pivoting possibilities. However, the interface and interactions to support this action are completely different. In Parallax, the user can click on a hyperlink representing the desired relation and the interface removes the original items and presents the new set of related items. For example, to pivot from publications to their respective authors, the user has to click on a link named "Authors". This interaction style is similar to hypertext browsing. In SeCo, the user has to activate an expansion mechanism in the menu, select the desired relation, and the tool adds the related items to the current focus without removing the original items. In both tools, despite the differences in the interface and interaction, the user can pivot from a set of items to a related set of items, which defines a many-to-many pivoting. Now, if we analyze the Tabulator tool, it presents the same interaction style of Parallax, where the user can click the relation and change the current focus, but it is only possible to pivot from a single item to another single item in a one-to-one pivoting.

Therefore, if we compare SeCo, Parallax, and Tabulator abstracting interaction and interface details against the pivoting action, we conclude that SeCo is as expressive as Parallax and more expressive than Tabulator.

The goal of the functional layer is to formally define a set of exploration actions that can be used as framework for analysis, design, and comparisons of exploration tools. The main requirement for such framework is to be expressive enough to describe at least the state-of-the-art exploration tools currently proposed in the literature.

In order to define the framework, we first carried out a literature survey for papers presenting Exploratory Search tools or Information Exploration environments and defined the main categories of the tools, namely Faceted Search Tools, Set-Oriented Browsers, Tabular Processors, and Relation Finders. For each tool, we analyzed their interfaces (when available), bibliography, and tutorials to synthesize a collated list of features. This list was then generalized into a set of operations and parameters. The tools analyzed were: Tabulator, Mspace (SCHRAEFEL et al., 2005), gfacet (HEIM; ZIEGLER; LOHMANN, 2008), /facet (HILDEBRAND; OSSENBRUGGEN; HARDMAN, 2006), tfacet (BRUNK; HEIM, 2011), Sewelis (FERRÉ; HERMANN, 2012), SemFacet (ARENAS, 2014), Parallel Faceted Browser (BUSCHBECK *et al.*, 2013), Visor (POPOV *et al.*, 2011), Parallax, Rhizomer (GARCÍA *et al.*, 2013), MusicPinta (DIMITROVA *et al.*, 2013), Relation Browser (ZHANG; MARCHIONINI, 2005), BrowseRDF (OREN; DELBRU; DECKER, 2006), Liquid Query (BOZZON, ALESSANDRO *et al.*, 2010), SearchComputing (BOZZON, A *et al.*, 2013), Tableau (HEER *et al.*, 2008), Explorator (ARAÚJO; SCHWABE, 2009) and its follow up RExplorator (COHEN; SCHWABE, 2012), Fusion (ARAUJO *et al.*, 2010), and RelFinder (HEIM; LOHMANN; STEGEMANN, 2010).

We present below a general description of the operations that describe at least the exploration actions of these state-of-the-art exploration tools:

- *Pivot(Items, Relation)*: receives a set of items and a relation and returns a set of related items. This operation describes the browsing possibilities found in the state-of-the-art;
- *Refine(Items*, *Filter)*: filters the current set of items to a set of items that matches the restrictions imposed by the explorer through the *Filter* parameter. The *Refine* operation along with *Pivot* and the set operations describe the majority of faceted search tools;
- *Group*(*Items*, *Relation)*: groups a set of items based on a relation, which can be defined either by the data model or by the user as a computed

relation. It is important to observe that this is an abstract definition, which can be specialized, for example, in a clustering operation, where the relation is a distance function;

- *Correlate(SourceItems, TargetItems)*: finds a set of relations that connects the two sets of items. This operation describes a category of exploration tools that discovers connections between items, such as, Fusion, RelFinder, and Visor;
- *Rank(Items, Fscore)*: ranks a set of items given a score function;
- *Map(Items, MappingFunction):* maps a set of items onto another set of items using a mapping function, where, the function is provided by the environment, such as counts and format and scale converters. This function is usually found in tabular processors for creating new columns (relations), hence, extrapolating the relations described by the data schema.

The need for a common agreement on the set of operations and its expressivity has already been verified in the literature (BOZZON, A *et al.*, 2013; FERRÉ; HERMANN, 2012; YOGEV *et al.*, 2012), however, it remains an open question. We do share the vision that achieving a complete framework, if possible, is a very hard task (WILSON, MAX L.; SCHRAEFEL; WHITE, 2009). Nonetheless, the claim of this work is to describe a representative set of state-of-the-art operations, rather than all possible actions.

The benefits of such framework are not restricted only to evaluation purposes. In our understanding, the exploration process is a sequence of applications of the exploration operations, where, the output of an operation is the input of the following, hence, forming functional compositions of operations. Therefore, the framework leverages accurate representation of exploration tasks that can be generalized and reused in future explorations, as we demonstrate through the case studies of chapter 5. A formalization of the exploration operations is presented in chapter 4.

3.2. Interaction/Interface

The Interaction/Interface layer is concerned with efficiently supporting the flow of execution of the actions defined by the functional layer with proper

interaction dialogues and interface controls. Based on the concepts of the functional layer, we can define general goals of the interface: (1) present items and relations without overloading the user with excessive information; (2) Allow the user to select an operator/composition and specify its parameters; (3) Allow the user to manage the task progress.

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" (SHNEIDERMAN, 1996) should be considered as a guideline. The visual information seeking mantra states that the user seeks for information in cycles, where in each cycle the user starts with a general description of the information items, adjusts the focus to some portion of the data of interest, filters uninteresting data, and selects a subset of items in order to ask for details.

The selection and application of exploration operations presents another challenge for the Interface/Interaction layer. As we illustrated for the *Pivot* action in the previous section, a single action can be articulated through many interaction patterns and interface controls. In addition, there is the possibility of devising interaction patterns and controls for certain combinations of operators that are usually executed together for specific types of tasks.

In order to illustrate the problem of interaction design for specific combinations of operations, lets analyze the case of more recent faceted search tools. The original faceted search paradigm was devised as an interaction style for refining data by applying restrictions to a set of dimensions, such as filtering publications by authors, publication years, and subjects (HEARST *et al.*, 2002). The main shortcoming of the first tools is the impossibility of changing the focus to filter items of different types. For example, it was not possible to pivot from publications to authors and then filter authors by affiliation, education, or research projects. More recent faceted search tools, such as /facet, gfacet, and Rhizomer, solved this problem by introducing the *Pivot* operation and an interaction that supports combinations of *Refine* and *Pivot* in order to allow refinements of multiple types of data. Although they present the same possibilities of combinations of *Refine* and *Pivot* their interfaces and interaction patterns are

completely different. Figure 8 and Figure 9 present screenshots of gfacet and Rhizomer.

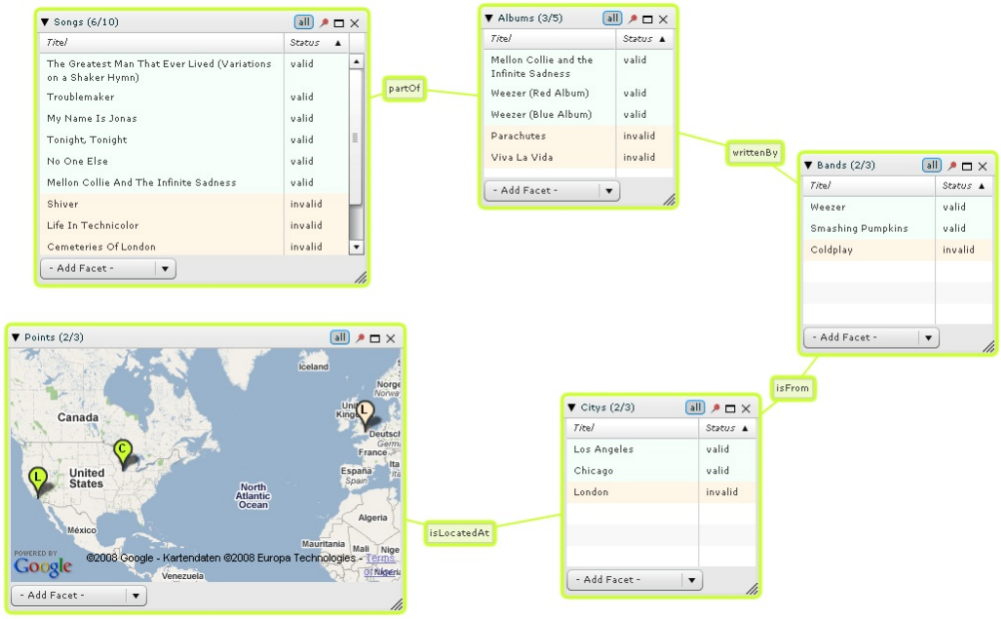

Figure 8 - gfacet screenshot

In gfacet, the exploration is represented as a graph of interrelated sets of items. In order to apply a *Refine*, the user should first select a relation, pivot to the related items and select the related item. When the user selects an item, all related sets are filtered accordingly. Therefore, in order to refine the data over a relation, the user needs to always pivot to the set of related items. On the other hand, Rhizomer does not impose the execution of a *Pivot* before a *Refine*, since the filtering relations and values are presented for selections along with the items to be filtered, as Figure 9 shows. Similar situations also occur for other combinations, such as combinations of *Refine* and *Rank* or *Group* and *Map*.

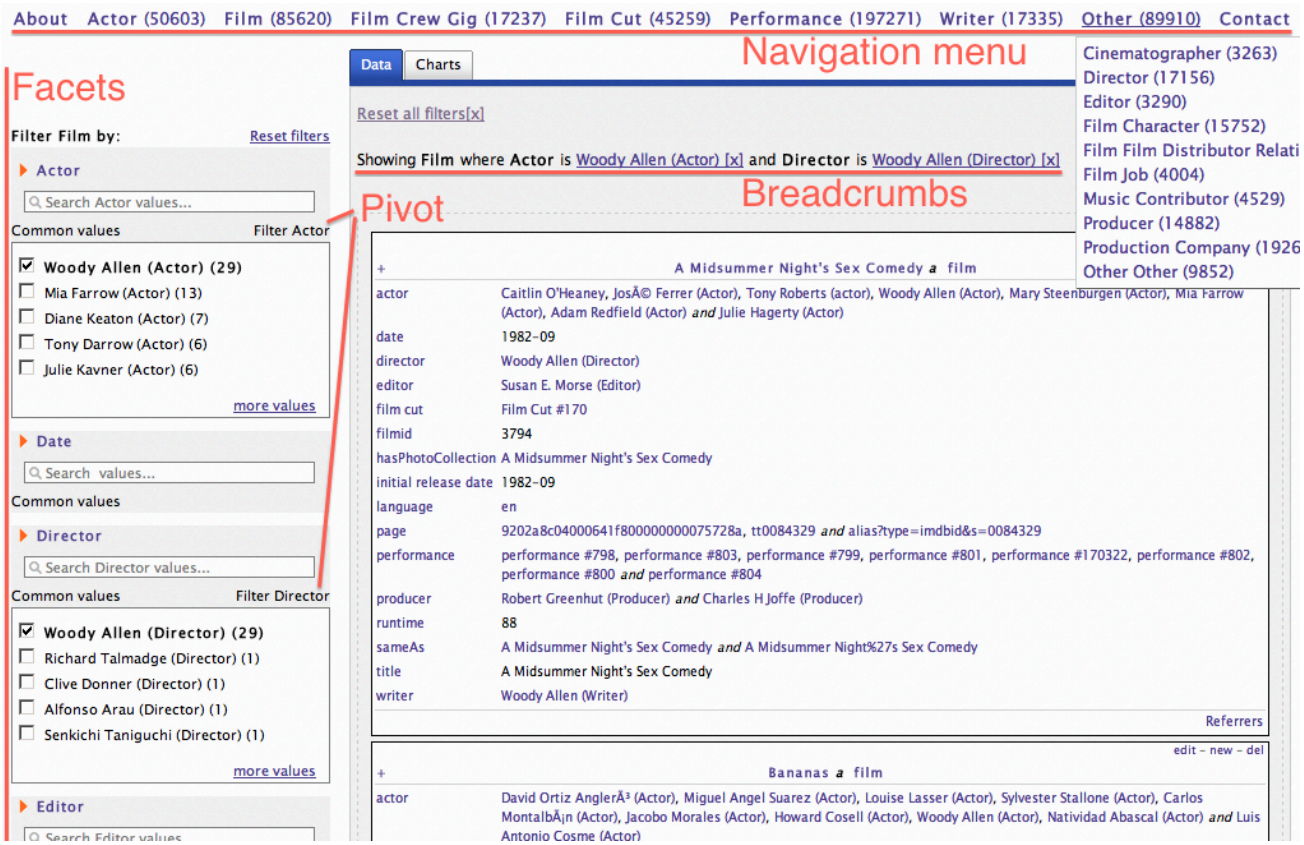

Figure 9 - Rhizomer screenshot presenting the filtering relations as facets and the pivoting controls (GARCÍA *et al.*, 2013)

A third Interface/Interaction concern is the task progress management. The interface should support, for example, the visualization of a history of the actions executed, undo and replay controls, sessions save and load actions, and annotations. The majority of exploration tools present an exploration trail containing the actions executed in the form of a line or a tree view with navigation controls. A detailed discussion of interface issues can be found in chapter 7.

3.3. Data Model

The user explores complex information spaces by executing operations in sequence over information items and relations. This leads to the issue of how to represent the items and relations that compose the information space so we can formally describe the semantics of each operation. A possible answer for this issue is to formalize the operations over some well-known data model defined in the state-of-the-art, such as the relational model (NIEMI; JÄRVELIN, 1983), the

RDF model[6] , and NoSQL models (CATTELL, 2011). However, exploration actions are not attached to a specific data model. For example, the semantics of the *Pivot* operation is independent of whether the items and relations are represented as sets of RDF triples in the form $<item_1, relation, item_2>$, tables and foreign-key relations, or document nestings in NoSQL document-oriented databases (CATTELL, 2011). In order to define the *Pivot* operation, we only need general characterizations of items, and relationships.

Another issue related to the selection of a proper data model is how the description of the operations maps to the operations described by the query languages of existing data models. There are some considerations that should be made in regard to this issue. First, our exploration model not only abstracts physical representations of data, but also auxiliary functions that support the execution of certain actions. For example, the grouping relation parameter of the *Group* action is not restricted to relations represented by the schema; it can also be a distance function that computes the degree of similarity between vector representations of two textual documents. In this way, we can also describe clustering as a specialization of the *Group* action. The same situation occurs with *Refine* and *Map*, where both the filters and the mapping functions are hotspots of the framework. Such flexibilization is the key to allow the description of explorations over unstructured data using the same operations. The difference is that, in case of unstructured data (e.g., text), the relations are approximations computed at runtime.

Another aspect that turns the direct mapping of our proposed operations into one of the extant data models and query languages is that the semantics of some actions are not easily mapped to a concise query statement. For example, the semantics of the operation *Correlate* can be mapped neither to a pure relational algebra statement, nor to SPARQL or NoSQL path queries. To the best of our knowledge, the closest approximation to find intermediary relations between items in a state-of-the-art query language is to issue at least n^2 recursive queries using the *$graphLookup* directive in MongoDB[7], where n is the amount of possible intermediary relations. Moreover, when analyzing the possibilities of

[6] https://www.w3.org/TR/WD-rdf-syntax-971002/
[7] https://docs.mongodb.com/manual/

compositions of operations, such as the application of refinements over results of correlations or groupings, the problem gets even worse.

Considering all the issues presented, we chose to define a simple data model of items and relations that is a generalization of the aforementioned data models and can be further mapped into each one of them. The definition of the data model and the operations can be found in chapter 4. Design and implementation issues are further described in chapter 7.

3.4. Related Works

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 (AMAR; STASKO, 2005; AMAR; EAGAN; STASKO, 2005; BREHMER; MUNZNER, 2013; CHI, 2000; SHNEIDERMAN, 1996). Chi's work (CHI, 2000) 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 (AMAR; EAGAN; STASKO, 2005) presents a taxonomy of analytic operations for describing visualization tasks containing, for example, "Cluster", "Filter", "Sort", and "Correlate" operations. The work in (BREHMER; MUNZNER, 2013) 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 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 (WHITE; ROTH, 2009). 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 (WILDEMUTH; FREUND, 2012), 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 (BOZZON, A *et al.*, 2013) 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 (ALAHMARI *et al.*, 2012) and (WILSON, MAX L.; SCHRAEFEL; WHITE, 2009) 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 design space 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.