Defining Interaction Design Patterns to Extract Knowledge from Big Data

Carlos Iñiguez-Jarrín[1,2], José Ignacio Panach[3], Oscar Pastor López[1]

[1]Research Center on Software Production Methods (PROS),
Universitat Politècnica de València. Camino Vera s/n. 46022, Valencia, Spain
[2]Departamento de Informática y Ciencias de la Computación,
Escuela Politécnica Nacional, Ladrón de Guevara E11-253, Quito, Ecuador
[3]Escola Tècnica Superior d'Enginyeria,
Departament d'Informàtica, Universitat de València
Avenida de la Universidad, s/n, 46100 Burjassot, Valencia, Spain

{ciniguez, opastor}@pros.upv.es, joigpana@uv.es

Abstract. The Big Data domain offers valuable opportunities to gain valuable knowledge. The User Interface (UI), the place where the user interacts to extract knowledge from data, must be adapted to address the domain complexities. Designing UIs for Big Data becomes a challenge that involves identifying and designing the user-data interaction implicated in the knowledge extraction. To design such an interaction, one widely used approach is design patterns. Design Patterns describe solutions to common interaction design problems. This paper proposes a set of patterns to design UIs aimed at extracting knowledge from the Big Data systems' data conceptual schemas. As a practical example, we apply the patterns to design UI's for the Diagnosis of Genetic Diseases domain since it is a clear case of extracting knowledge from a complex set of genetic data. Our patterns provide valuable design guidelines for Big Data UIs.

Keywords: User Interfaces, interaction patterns, Big Data.

1 Introduction

Extracting knowledge from Big Data is not a trivial task and usually involves interacting with the data by identifying, combining and managing multiple and heterogeneous data sources as well as constructing advanced analysis models to predict outcomes. This task is performed at the User Interface (UI)—the contact point between user and data. To design UIs for Big Data domain, software designers should know the data consumption challenges in this domain, understand the needs of the direct beneficiaries of the information and formulate solutions to design the UI.

The Interaction Design Patterns approach is commonly used to design UIs. An Interaction Design Pattern (IDP) deals with an interaction problem in the UI design and

provide a design solution to solve it. The interaction problems in the Big Data analysis are different from the traditional data analysis ones. How to interact with the data when they surpass any of three dimensions volume, velocity, and variety [1] is a problem to address in Big Data and the literature do not fit such Big Data interaction need; therefore, identifying suitable IDPs that help the designers to develop interactive Big Data UIs is a challenge.

We aim to define a set of IDPs for extracting knowledge from Big Data. To do that, we a) analyze several real Big Data use cases available in the literature, b) identify the challenges posed by the domain, and c) derive IDPs as solutions to such challenges. Each IDP describes in detail the user-data interaction, referencing the data schema (conceptual model) that the interactive system supports and highlighting the effect that the dimensions of volume, velocity, and variety produce on the interaction.

We illustrate the IDPs by applying them to the *Diagnosis of Genetic Diseases* (DGD) domain, a concrete Big Data example. In this domain, the researchers extract knowledge by contrasting huge volumes of genetic data with information from data sources of heterogeneous formats [2]. Several tools have been developed to support the researchers in the genetic data analysis [3]. However, the lack of intuitive and interactive-usable mechanisms of such tools converts the analysis activity into a complex and time-consuming one. That is why this domain is the perfect setting to apply the defined IDPs. We illustrate the IDPs through UIs designed for analyzing genetic diseases and highlight the benefits of using such UIs.

This paper is organized as follows: Section 2 discusses several endeavors aimed to define IDPs. Section 3 discusses relevant concepts in the DGD, the domain where we will illustrate our patterns. Section 4 describes the Big Data challenges upon which the IDPs have been derived. Section 5 presents the classification of the proposed IDPs and how they have been applied to the illustrative example. Finally, section 6 discusses the conclusions and outlines future work.

2 Related Works

The *pattern* term comes from the design of buildings and architectural planning [4]. Later, the pattern term was adopted by HCI and Software Engineering disciplines. Especially in the UI design context, the patterns emerged to solve interaction design problems, hence its name "*interaction design patterns*".

There is a wide range of IDPs collections, available in articles, web sites, and books addressing several platforms and domains. Tidwell' book [5] describes a pattern collection to deal with common design problems of the web, desktop, mobile, social media UIs. For example, the "Showing Complex Data" chapter deals with design problems to represent large and complex data sets. Duyne et al. [6] present a pattern collection, organized into 13 groups, to design websites pointed to several domains (e.g., government, e-commerce, educational). The "K. Making Navigation Easy" group, for example, contains 14 patterns to make the web navigation easy to understand and easy to find by applying techniques for organizing and displaying navigational elements. In the Information Retrieval domain, Schmettow's article [7]

exposes a pattern language containing 10 patterns for designing UIs in this domain. Other patterns dealing with massive data sets have been presented by the industry such as Expero (an enterprise focused on the user experience for complex domains) that exposes through webinar[1] a set of patterns to solve design interaction problems such as How to configure a complex data table to view the status at a glance? Or When to apply column filtering to display what user wants to see?

In the Big Data domain, the published literature about IDPs is limited, and the little existing literature describes the problems rather than providing solutions. In particular, the problems related to *data consumption* (i.e., the way in which users extract knowledge from data) are documented in Big Data's use cases ([8], [9], [10]), however, there is no formal proposals to solve them.

In relation to how to apply the patterns to design UIs, as far we know, there is no formal method or standard recipe. However, *Situational Method Engineering* (SME) [11] becomes a potential framework for formulating a pattern-based UI design method. SME aims to create a method adapted to a specific situation by harmonizing fragments of existing standard methods. The Big Data UI design is a specific situation where the patterns, which form the UI, behave as fragments that can be selected from situational factors and data characteristics.

At the user interface level, Big Data interaction problems have been little studied. Addressing these problems involves considering the Big Data dimensions and the inherent connection between the UI and what happens behind it. In this work, we will analyze the Big Data's use cases to define the interaction involved in the knowledge extraction and then we will suggest IDPs as solutions to design UIs for extracting knowledge from Big Data.

3 The Diagnosis of Genetic Diseases (DGD)

Differences between humans are registered in the genome under the name of *"genetic variation*". Genetic variations not only represent different physical traits among humans but can also be the potential cause of genetic diseases (e.g., Alzheimer, Neuroblastoma). The DGD aims to extract knowledge from genetic data to diagnose a certain disease by identifying the disease-causing genetic variations and it represents a complex and time-consuming data comparison process. The researchers use web browsers provided by public biological databases (e.g., PubMed, ClinVar, dbSNP, and other databases maintained by the NCBI[2]) to search for and identify the diseases related to each of the genetic variations of a sample, which are commonly stored in large digital files (e.g., *VCF file*[3]). The main concepts and their relationships surrounding the DGD are described in an earlier work [12] through a conceptual model (CM) upon which an application for DGD, named GenDomus, was designed.

---

1 https://www.experoinc.com/post/ui-design-patterns-for-navigating-complex-data-sets-online-seminar.

2 *National Center for Biotechnology Information.* https://www.ncbi.nlm.nih.gov/guide/all/

3 The Variant Call Format (VCF) Version 4.2 Specification. https://samtools.github.io/hts-specs/VCFv4.2.pdf

DGD is a clear example of Big Data. Table 1 shows the special data characteristics of this domain in terms of Big Data dimensions.

Table 1. DGD data characteristics in terms of Big Data dimensions.

VCF files	Volume	The size of a VCF file ranges from gigabytes to terabytes. Frequently, more than one file is required to perform the data analysis.
	Variety	The content format of this text file is semi-structured. Each genetic variation in this file is described in one row and several tab-separated text columns; however, several different data can be stored in the same column.
	Velocity	VCF files are processed in batch mode.
Biological databases	Volume	Especially, PubMed, the database most frequented by researchers, contains more than 28 million citations of biomedical literature, as reported on its website[4].
	Variety	Biomedical literature contained in NCBI biological databases is available as web text, PDF documents, images, videos, or XML format.
	Velocity	The data from biological databases can be retrieved on demand by using available APIs or FTP service and processed in batch mode.

A systematic way to perform the DGD is the SILE methodology [13], which consists of four levels: **S**earch (the researchers search for external biological databases and select those that serve the analysis purposes), **I**dentification (for each biological database, the researchers identify the content suitable for the analysis), **L**oad (the VCF files and the biological databases content are loaded into a central repository) and **E**xploitation (the researchers explore, operate, analyze, and extract knowledge from genetic data by drawing conclusions from them).

In the next sections, we will illustrate how the proposed patterns, embedded into UIs, match with the SILE methodology levels and deal with the interaction difficulties immersed in identifying disease-causing genetic variations.

4 Big Data Challenges in Knowledge Extraction

Our two initial steps in defining IDPs in the Big Data are: to analyze several real Big Data's use cases and to identify clearly the challenges related to the *data consumption* issues (i.e., browsing, exploring, and visualizing the data). We analyze real Big Data's use cases since they become ideas arising from real environments to address a specific need, as explicitly described by IBM [8]. We studied 27 Big Data use cases (5 IBM's Big Data use cases [8], 7 Pentaho's use cases [10], 10 use cases exposed by Laskowski [14], and 5 use cases reported by Datamer [9]) from different domains and industrial sectors (e.g., online stores, financial services, security, real state, pharmaceutical, energy management, IoT and telecommunications service). From the set of Big Data use cases, we identified the common challenges related to the knowledge extraction and summarized them as follow:

- *Challenge 1*: **Enhance Data Discovery**. – Need for novel mechanisms to navigate and explore structured and unstructured data sources. These mechanisms should consider the large amount of data and the diversity of data sources, the provenance

[4] https://www.ncbi.nlm.nih.gov/pubmed/

(e.g., data within and outside the limits of what you have). This challenge is related to the use cases in [8].

- *Challenge 2*: **Enlarge visualization**. – Need for advanced visualizations and powerful interactive dashboards to present, in a single UI, a complete view about an entity of interest. The visualization should consider different perspectives to represent the data (e.g., data charts) and support static data as well as data in motion (i.e., such as audio, video, social media). This challenge is related to the use cases in [8], [10], [9].
- *Challenge 3*: **Perform data analysis operations.** – Need for intuitive mechanisms to operate the data to find differences and similarities between different datasets. Data operations should consider the different data types (e.g., relational data, machine data, social media data) as well as the state of the data (static data or data in motion). This challenge is related to the use cases in [8], [14], [9].
- *Challenge 4*: **Contextualize data by augmenting it**. – Current data analysis relies on the data warehouses content; therefore, the conclusions drawn from the analysis are limited to such data warehouse content. To draw better conclusions, users require enriching the data warehouses with information in context (i.e., place, social space, time) from internal or external multi-structured data sources. This challenge is related to the use cases in [10], [14], [9].

The Big Data challenges are common to all domains dealing with large volumes of heterogeneous data, and the genomic domain is one of them. Therefore, the challenges will serve as a starting point to derive IDPs for extracting knowledge from genetic data.

5 Interaction Design Patterns (IDP)

Based on our context, we define an IDP as *a proven solution to a recurring interaction design problem in the knowledge extraction from Big Data.* Where "*interaction design problem*" refers to the difficulty of designing the user-data dialog when extracting knowledge from data. Therefore, the answer to "*how*" the dialog can be performed becomes an IDP that can be implemented into a UI, allowing the user to complete his/her task. For example, when designing the UI for purchasing an airline ticket, one of the design problems we must face is: *How can the user specify the departure and return dates?* The proven solution to such a design problem is an IDP which has already been documented by other authors (e.g., the Date Selector pattern[5] from the Welie.com catalog).

A UI is an individual piece of presentation containing visual components known as *widgets* (e.g., list box, push button) which can be manipulated by the user to perform an interactive task. Designing UIs is a difficult task since the aesthetic and behavior aspects should be considered as well as the strong dependency with the platform derived from the use of concrete widgets. To avoid such difficulties we adopt a high-

[5] Data Selector pattern. http://www.welie.com/patterns/showPattern.php?patternID=date-selector

level perspective by focusing on the abstract and generalist instead of the concrete and specific. To this end, the concept of **interaction units** (IU)[15] becomes a useful perspective since it defines an individual piece of presentation by abstracting both the presentation and behavior from the UI. The IU represents the "*what*"— the task or set of tasks to be performed by the user—and contains several **interaction patterns** representing the "*how*" such tasks can be performed. Indeed, an IU encloses several interaction patterns aimed to achieve a common goal. The interaction patterns range from the simplest to the most complex interaction. **Elemental patterns** represent the atomic interactions becoming the building blocks of the interaction and can come together to form more complex compound patterns called **composite patterns**. Using the elemental patterns proposed in this paper implies the existence of an underlying CM that represents the data abstractly since such patterns work with the information represented in conceptual schemas. The CM is specific to the problem to solve, in our example, we use the DGD model [12] as illustrative study.

From the implementation point of view, an IU becomes the form or web page with which the user interacts [15]. A standard website for purchasing airline tickets, for example, contains at least three IUs represented by three web forms such as "*Book flights*" to specify departure dates, "*Choose flights*" to specify the available flights and "*Purchasing flights*" to pay for tickets. The user interacts with each IU to achieve his/her goal: "*To buy an air ticket*".

In the *Book flights* web form, the "how the user can specify the departure and return dates" could be solved through an elemental pattern whereas in the "*Purchasing flights*" web form, the "how to deal with the pay process*"* could use a composite pattern containing two elemental patterns: *The input data for the credit card information* and *the payment gateway* that informs whether the payment has been approved or denied.

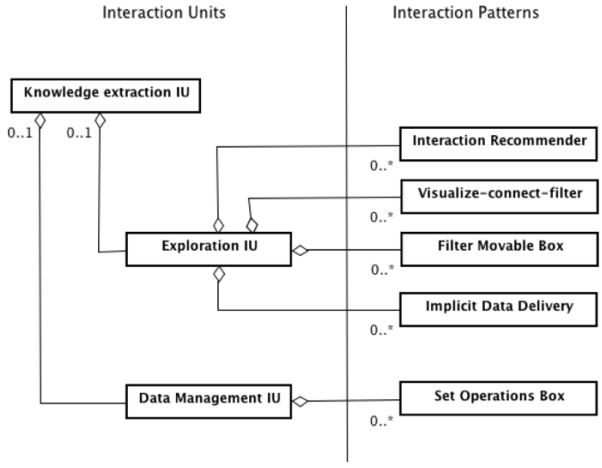

Fig. 1. Overview of the interaction units (IU) and interaction patterns for extracting knowledge from Big Data

Based on the identified Big Data challenges, we identify the IUs and patterns needed to extract knowledge from the Big Data domain, as shown in Figure 1. In this section, we will define each IU and interaction pattern and demonstrate how they can be applied through an illustrative example: The Diagnosis of Genetic Diseases.

5.1 Interaction Units

From Figure 1, the data environment for extracting knowledge is made up of three (3) IUs, defined as follow:

Knowledge extraction. - It acts as an IU container by encapsulating the set of indispensable IUs for extracting knowledge. This IU is related to all Big Data challenges (i.e., challenge 1 to 4) since it provides access to the IUs involved in the knowledge extraction space, as shown in Figure 2 or 3 for our illustrative example (DGD).

Figure 2, for example, shows the "navigation tabs" to navigate to the Data Management and Exploration UIs. The implicit top-level UI containing the navigation structure becomes the implementation of this IU.

Data Management. – It searches for and identifies the available data sources and highly related to the subject under study. This IU is related to challenge 3.

Figure 2 shows the UI web that implements this UI for our illustrative example (DGD). Because not all the content of the data sources is useful for diagnosis, the content must be verified and selected before being loaded and thus safeguard the performance of the application. Therefore, the user can work with a *comparison matrix* (Figure 2b) where the data sources and genetic samples available in the "Data Catalog" panel can be added and compared in memory (Figure 2a). Crossing of genetic samples with the content of the data sources reveals similarities or significant differences that are useful to determine the appropriate content of the sources to be loaded for analysis purposes. The red "Set operation" red box represents an IDP involved in this UI that will be explained in detail in the next section.

Exploration. – It allows the user to explore, visualize, analyze and understand the data, make relevant annotations and share insights to draw conclusions. This IU is related to the challenges 1, 2 and 3.

Figure 3 shows the web UI design that implements this IU for our illustrative example (DGD). Through this UI, the user prioritizes the genetic variations and genes causing a certain disease by exploring and filtering the data, as well as visualizing the data distribution from different perspectives. This UI is made up of three panels located on the left, top and bottom of the UI, respectively. The left panel, hosting two subpanels, aims to keep the user aware of the data analysis interactions performed and provide contextual information about a certain topic of interest. The top panel (“Navigator” panel) allows to visualize the data from several perspectives and manipulate the data in a direct and intuitive way. The bottom panel displays the resulting data from the filter conditions applied in the top panel. The IDPs involved in this UI will be explained in the next section.

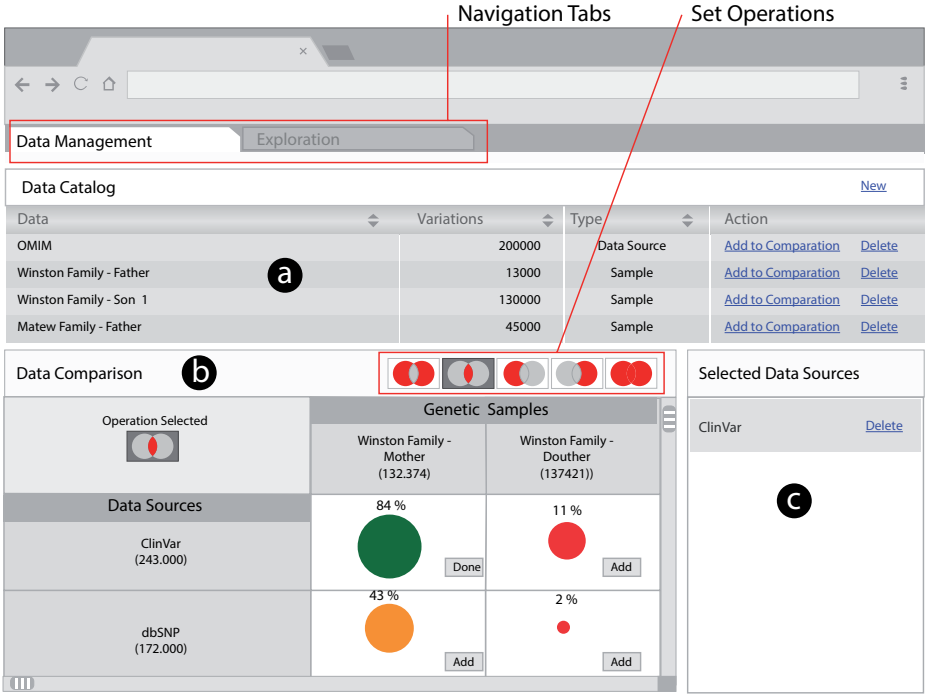

Fig. 2. Design of data management user interface. Source: The authors

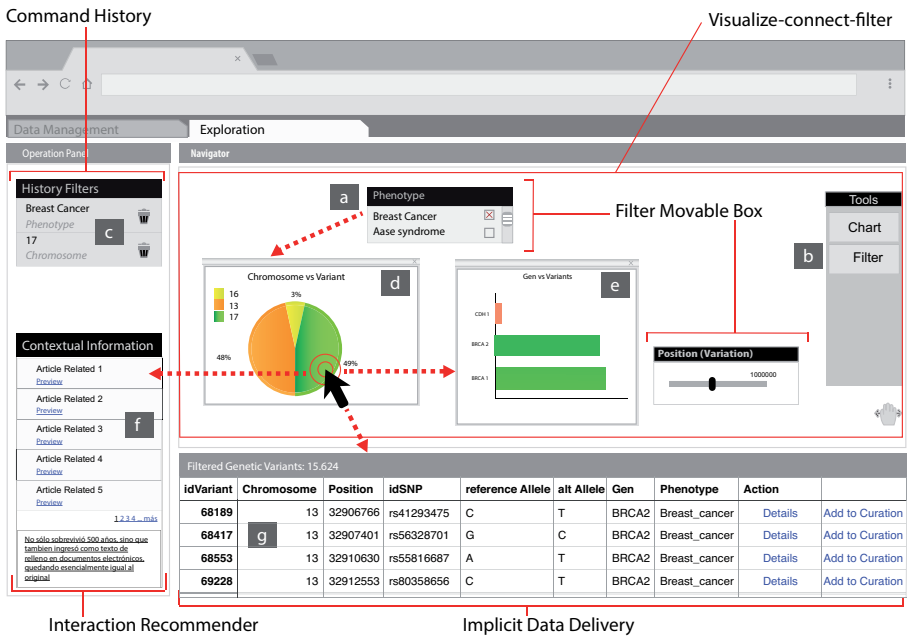

Fig. 3. Design of exploration user interface. Source: The authors.

5.2 Interaction Patterns

In this section, we define the IDPs stated in Figure 1 by using a five-parts template (*name*, *context*, *problem*, *solution, why,* and *example*) which was defined as a common denominator between other analyzed templates ([7], [5]). Especially the "*example*" part of the template describes how the pattern has been applied to the DGD, our illustrative example, by referring to CM [12] of the DGD. What challenge solves each pattern and what level of SILE methodology is affected by each pattern are showed at the end of this section.

Name: Implicit Data Delivery

Context: User interfaces showing vast volumes of data present performance problems. The user navigates across the dataset by visiting part by part like a person navigates across the pages of a huge book.

Problem: Users experience delays when scrolling through large data volumes. How to improve the user experience when showing and navigating through large data volumes?

Solution: Allow the user to navigate freely across the data set. Loading and delivering the data as the user navigates through a data set of interests. The data is loaded only when the user expresses implicitly her intention to move across them (e.g., navigate freely by an entire data table), rather an explicit intention of the user (e.g., paging control). This solution is applicable to retrieve any data object from the underlying data schema.

Why: Users need to visualize and navigate the data in a fluid manner, even when the data volume grows exponentially to terabytes or petabytes and the sources that provide them are scattered and with different structures (i.e., structured or unstructured). Navigating and viewing the entire data set is very expensive in terms of performance. The implicit on-demand data delivery is an adequate alternative since it motivates the user to discover the unknown data set by visualizing it and navigating it through each of its parts, focusing on one area of interest at a time. The navigation and visualization status is updated according to the intention to consume the data, making the data recovery transparent to the user. This pattern takes advantage of the performance and computational processing by loading the data on demand. Thus, the UI is not overloaded with data.

Example: In the Identification and Exploitation SILE's levels for the DGD, the user scrolls through large sets of genetic variations (represented by the *Variation* class in the CM). Figure 3g shows a data table containing more than 10.000 genetic variations records to be explored (a number easily reached in the genetic test, even when several data filter conditions have been applied), of which only 4 are visible. Instead of using a paging control to navigate the data table, the table implements the "Implicit Data Delivery" pattern allowing the user to fluidly navigate the content. When the user scrolls freely the table, only the section of genetic variations of interest is implicitly requested, loaded, and displayed. Therefore, the user can scroll from the first record to the thousandth record through a single mouse scroll movement without overloading the UI with data.

Name: Filter Movable Box

Context: The user segregates large volume of data by using predefined filter options which are generally placed in a restrictive area of the UI. To filter and visualize the effects in the data, the user moves to the predefined filter area, then apply the filter and finally, move again to the data chart to visualize the results. The time of response is affected by the required performance to process the large volume of data.

Problem: How can the user define filter conditions and apply them in a fluid way within the analysis space?

Solution: Allow the user to define data filter conditions from any intrinsic object attribute in the underlying data schema. Of course, the representation aspects and filter operations available in each filter box will depend on the object attribute's data type. A numeric data type, for example, can be well represented by a "single slider" component to select a certain value or a "multiple slider" component to determine a range of values. Likewise, filter operations such as *greater than* (>), *equal to* (=) can be included to express the data filter condition. The user is free to move the filter box across the canvas and to place them wherever he/she need it.

Why: The "pre-defined" filters limit the data filtering to specific criteria of the domain. What if we need to extend the set of filters? This pattern allows the user to create filter options from any attribute from the underlying data schema under analysis. In the front-end, the filter boxes can be moved and placed next to data charts allowing the user to filter the data and to look directly at the effect caused, thus, users avoid moving unnecessarily between the parts of the UI (data charts and the filter options usually predefined and strictly located in a specific area within the UI) to look at the results of the data filtering. In the back-end, performance issues need to be considered to process the large data volume since high performance to index and process the data on the fly is required when filtering the data. Node-based databases engines are suitable to store large volume of structured and structured data; however, the indexation of data is an issue to be considered when filtering the data.

Example: In the Identification and Exploitation SILE's levels for the DGD, a common task is to filter the genetic variations. Figure 3a shows this pattern implemented to assist the user to perform such task. From the CM, the user has selected the *name* and *position* attributes (from *Phenotype* and *Variation* classes respectively) to create the *Phenotype* and *Position* filter boxes. So, when the user selects the "Breast Cancer" as disease name in the Phenotype filter box, the visual data components (i.e., data charts, data table) refresh their state by showing the updated distribution of variations matching the Breast Cancer disease. Likewise, by using the Position filter box, the user filters the genetic variations ranging from 10.000 to 20.000 positions into the chromosome.

Name: Set Operation Box

Context: Users compare two or more data sets to find similarities or differences between them. Data query languages rely on set operations to compare data sets. Such languages are difficult to use for non-technical users.

Problem: How can the user perform set operations on data in an intuitive way?

Solution: Provide the user a set operations kit (i.e., union, intersection, complement, difference) to compare data sets sharing at least one common attribute of the

data schema. The user must select as input, at least two data sets from the data schema, and then select the set operation to be performed. Because of comparison, a new data set is created that can be reused in further comparisons. In the Big Data domain, the set operations must be supported by a sound technological architecture with a high-performance data processing because of a large amount of existent data.

Why: The user identifies valuable information by comparing several data sets. Often. This pattern allows the user to operate the data intuitively avoiding using and understand complex query languages and relying on technical users to extract knowledge from the data. The variety and volume of data are two Big Data dimensions committed in this pattern. The first involves the hardware performance needed to integrate dispersed sources with different formats that participate in the operation. The second involves increasing memory and storage to store the data produced by the operations performed. A hardware architecture with a flexible vertical and horizontal scaling support is appreciated for the implementation of this pattern.

Example: In the Search and Identification SILE's levels for the DGD, one of the key issues is to select the adequate data sources for the analysis. The reliability of the findings depends on large extent on the data used in the analysis. Not all the available data is useful for the data analysis. Selecting the adequate data sources, those containing the most number of variations matching with the variations in the sample under study, ensures valuable and accurate conclusions. Figure 2 shows this pattern applied to identify the adequate data sources for DGD. From the Data Catalog (Figure 2a), the user adds the data sources (represented by *Data Source* class in the CM) to the comparative matrix (Figure 2b) in the Data Comparison panel. The panel contains the "set operation" box located in the top-right corner. The intersection, the set operation selected in this case, is applied to each data source located in the first column and to each genetic sample located in the first row. The number of genetic variations matching between the data sources and the samples is shown graphically through a circle of variable area along with its value expressed as a percentage. The user selects the data sources containing the highest degree of concurrence with the samples. Finally, the selected data sources are added to the "Selected Data Sources" panel.

Name: Visualize-connect-filter

Context: Users use data charts (e.g., bar, lines, maps) to understand the data, even more, if it comes in enormous data volumes. A chart is an isolated view that shows how the data behaves respect to a particular point of view. Working with multiple data charts makes possible to show the behavior of the entire data set. However, working with several charts at the same time is complicated. If a filter condition is applied to the entire data set, each data chart must be manually updated to present the new behavior.

Problem: How can the user visualize the multiple changes in the data behavior caused by a simple filtered data?

Solution: Allow the user to orchestrate all data charts to visualize, from different perspectives, how the data behaves against a certain condition. Provide a canvas where the user is free to place interactive data charts and organize them according to their needs. For each chart, the user sets the data objects, from the underlying data schema, to be displayed. To define the behavior between the charts, the user can de-

fine links between the data objects that populate the charts. In this way, an interaction performed (e.g., filter) on a chart' sector will automatically affect the display state (behavior) of the charts linked to it.

Why: This pattern allows users to explore and filter the data visually, intuitively and interactively, focusing on the details while maintaining the overview of the big picture. Combining this pattern with the Command History pattern [5] that shows the list of events performed, the user remains aware of the traceability of the actions. The Big Data dimensions of volume, variety, and velocity are directly related to this pattern. In many cases, visualizing the large volume of data requires an architecture that supports GPU accelerated visualization. Display components must support the variety of content formats (images, text, video, sound) including graphics that support the visualization of multiple variables and various organizational models (e.g., hierarchical, tabular, geographic, linear data, node networks interconnected). The connection between interactive graphics becomes an alternative to integrate dispersed and heterogeneous data sources in format. When graphs are connected through semantically similar data attributes, the underlying data sources that feed each graph are implicitly integrated. Depending on the velocity of data consumption (i.e., real time, or batch mode), visualization components can range from 2D distribution data charts to complex real-time monitoring charts that allow configuring alerts to react to events. Contrary to data processed in batch mode, real-time data imposes high performance on rendering the visualization implying the need for cloud rendering software or an underlying architecture supporting hardware-accelerated interactive data visualization.

Example: Figure 3 shows this pattern implemented for the Exploitation SILE' level for the DGD where the user can filter intuitively and visualize the data. By selecting the *Chromosome* and *Variation* classes from the CM, the user has set the Pie Chart (Figure 3d) entitled as "Chromosome vs Variant". Similarly, the *Genotype* and *Variation* classes from the CM have been used to set the Bar chart (Figure 3e) entitled as "Genes vs Variations". The dashed lines indicate the dynamism when filtering and exploring the data. When the user "*click*" the sector corresponding to the chromosome 17 in the "Chromosome vs Variant" chart (arrow pointer), the "Genes vs Variations" chart refreshes its content by presenting the genes contained within the chromosome 17 (i.e., CDH, BRCA2, BRCA1), and how they are distributed according to the number of genetic variations. In addition, each filter interaction is registered into the "History Filters" panel (i.e., an implementation of the Command History pattern).

Name: Interaction Recommender

Context: Users need to be supported when navigating across the voluminous and heterogeneous set of data. By interacting with the data, the user requires relevant information (e.g., events, people, places, things) relatives to the data of interest.

Problem: Each user-data interaction expresses the user needs in an indirect way. How can the user be provided with knowledge obtained from such interactions?

Solution: Store the interactions performed by the users together with the involved data schema' elements and the related contextual information. For each meaningful interaction performed by the user, the application identifies similar stored interactions, gathers contextual information and defines alternatives exploration ways based

on the data schema. The application uses the information to support the user across the data navigation.

Why: The human capacity to figure out all relationships in the large volumes of data is limited. This pattern uses the user-data interaction (e.g., select, filter), compare it with existent ones to obtain contextual information which is made available to user as suggestions to enhance his/her data navigation. The variety and volume Big Data dimensions affect the user-data interaction. The underlying infrastructure must to consider automated and scalable storage to store the large volume of interactions, as well as, the mechanisms to analytically process the interactions (e.g., classification and clustering algorithms), retrieve the interaction-related contextual information (e.g., ETL or microservices), and display the contextual information related to the interactions considering the different content formats (e.g., images, text, video, audio).

Example: This pattern is useful for the Exploitation SILE' level for the DGD. From the interactions performed by the user in the data exploration, the pattern suggests relevant information as illustrated in the Contextual Information panel (Figure 3f). By selecting a specific genetic variation from the data table (Figure 3g), the pattern recommends useful resources such as clinical reports, research papers or studies which are related to the selected genetic variation. All resources obtained come from previous interactions similar to the current interaction. In addition, to motivate the exploration of the data, the system identifies the conceptual model elements closely related to the genetic variation (Chromosome, Phenotype, Gen) and suggests alternative contents under the template: *People who searched for "A", searched for "B" too.*

The correspondence between the described patterns with the Big Data challenges and the levels of the SILE methodology mentioned in our illustrative example is shown in Table 2. The Load (L) level of SILE methodology is not covered by the interaction patterns because of this level is pointed to data processing data level instead of front-end level.

Table 2. Interaction patterns vs Big Data challenges and SILE methodology levels.

Interaction Pattern	Interaction Unit	Big Data Challenge				SILE level			
		1	2	3	4	S	I	L	E
Filter Movable Box	Exploration	X					X		X
Set Operation Box	Data Management	X		X		X	X		
Visualize-connect-filter	Exploration		X						X
Intraction Recommender	Exploration		X		X				X
Implicit Data Delivery	Exploration	X	X				X		X

6 Conclusions and Future Works

In this document, we define interaction patterns for designing user interfaces for extracting knowledge from Big Data. The proposed patterns were derived from data consumption challenges identified from the study of several real use cases of Big

Data. The patterns to extract knowledge from Big Data were illustrated through the design of two user interfaces for diagnosing genetic diseases. The first one, to select the right data sources for the data analysis through an intuitive mechanism based on set operations to compare data. The last one, to explore the genetic data by incorporating recommendation mechanisms and interactive data filtering.

Historically, the patterns have been applied in software engineering to increase the performance in the development of applications, as well as the improvement of their quality. In this sense, the application of the proposed interaction patterns aims to accelerate the design of Big Data user interfaces that incorporate quality attributes such as efficiency and usability and that promote productivity in the extraction of knowledge.

From the experience in the design and implementation of GenDomus' UIs for the genetic data analysis, we have identified the interaction needs of the user and the data characteristics involved, as well as, the available solutions to face the user needs and the difficulties to implement them. The proposed patterns gather all those experiences and provide useful recommendations for consuming genetic data considering the dispersion of genetic sources, the type and the volume of data. This experience can be extended perfectly to the Big Data domain.

Once the patterns have been defined, our next step in the research line is to refine, implement, evaluate the proposed patterns and propose a methodology to apply the patterns. Although the defined patterns do not solve all the interaction problems in Big Data, we consider that the patterns presented here become the initiative to create a catalog of interaction patterns to design user interfaces oriented to the extraction of knowledge from Big Data.

Acknowledgments

The authors thank the members of the PROS Center's Genome group for fruitful discussions. In addition, it is also important to highlight that Secretaría Nacional de Educación, Ciencia y Tecnología (SENESCYT) and Escuela Politécnica Nacional from Ecuador have supported this work. This project also has the support of Generalitat Valenciana through project IDEO (PROMETEOII/2014/039) and Spanish Ministry of Science and Innovation through project DataME (ref: TIN2016-80811-P).

References

1. D. J. Power, *Decision Support Systems V – Big Data Analytics for Decision Making*, vol. 216. Cham: Springer International Publishing, 2015.
2. Genetic Alliance, "Capítulo 2, Diagnóstico de una enfermedad genética," 2009. https://www.ncbi.nlm.nih.gov/books/NBK132200/.
3. S. Pabinger *et al.*, "A survey of tools for variant analysis of next-generation genome sequencing data," *Brief. Bioinform.*, vol. 15, no. 2, pp. 256–278, Mar. 2014, DOI: 10.1093/bib/bbs086.
4. J. O. Borchers, "Pattern approach to interaction design," in *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, DIS*, 2000, pp. 369–378.

5. J. Tidwell, *Designing interfaces*, vol. XXXIII, no. 2. O'Reilly Media, 2012.
6. D. K. Van Duyne, J. A. Landay, and J. I. Hong, *The design of sites : patterns, principles, and processes for crafting a customer-centered Web experience*. Addison-Wesley, 2003.
7. M. Schmettow, "User interaction design patterns for information retrieval," *Eur. 2006*, pp. 489–512, 2006.
8. "IBM big data use cases – What is a big data use case and how to get started - Exploration." http://www-01.ibm.com/software/data/bigdata/use-cases.html.
9. Datamer e-book, "Top Five High-Impact Use Cases for Big Data Analytics," 2016. https://www.datameer.com/pdf/eBook-Top-Five-High-Impact-UseCases-for-Big-Data-Analytics.pdf.
10. "Big Data Uses Cases | Pentaho." http://www.pentaho.com/big-data-use-cases.
11. B. Henderson-Sellers and J. Ralyté, "Situational method engineering: state-of-the-art review," *J. Univers. Comput. Sci.*, 2010.
12. C. Iñiguez-Jarrin, A. García, J. F. Reyes, and O. Pastor, "GenDomus: Interactive and Collaboration Mechanisms for Diagnosing Genetic Diseases," in *{ENASE} 2017 - Proceedings of the 12th International Conference on Evaluation of Novel Approaches to Software Engineering, Porto, Portugal, April 28-29, 2017.*, 2017, pp. 91–102, DOI:10.5220/0006324000910102.
13. J. F. R. Román and Ó. P. López, "Use of GeIS for Early Diagnosis of Alcohol Sensitivity," in *Proceedings of the BIOSTEC2016*, 2016, pp. 284–289, DOI:10.5220/0005822902840289.
14. N. Laskowski, "Ten big data case studies in a nutshell." http://searchcio.techtarget.com/opinion/Ten-big-data-case-studies-in-a-nutshell.
15. P. J. Molina, S. Meliá, and O. Pastor, "Just-ui: A user interface specification model," in *Computer-Aided Design of User Interfaces III*, Springer, 2002, pp. 63–74.