OpenedEyes: A web standards-based generic framework for multidimensional information visualization

Caio Sacramento de Britto Almeida, *Computer Science Department, Federal University of Bahia - Brazil*
and Antônio L. Apolinário Jr., *Computer Science Department, Federal University of Bahia - Brazil*

***Abstract*—A huge amount of digital data is produced everyday, which led us to a situation called "data pollution". A big challenge is to extract knowledge from all this data, but many times it is presented in a way that is not friendly for a human being. Information visualization tries to solve this problem by representing datasets graphically and so making easier the task of comprehending a large dataset. Many of the available information visualization tools focus on desktop, require advanced skills from the user or are based on proprietary technologies. This work proposes an open source framework for information visualization called OPENEDEYES, written in pure JavaScript and based on web standards. The framework implements some interactive visualization tools that run on any modern web browser, which shows its the flexibility and extensibility. A case study using a real dataset was made for an empirical framework evaluation. Finally, a new visualization technique was proposed, a variation of the classic parallel coordinates technique, optimized for the web environment and for large datasets.**

***Keywords*—information visualization, framework, web standards.**

I. INTRODUCTION

THE evolution of communication and information technologies allowed to store and produce huge amounts of data. Researchers from the Berkeley University estimate that every year 1 exabyte of data is produced, most of them available in digital format [1]. While those datasets promise to carry valuable information, it's not a trivial task to find it out.

Many tools were developed to help on data analysis. A visual approach involves the user directly, is intuitive and can handle heterogeneous data, which is an advantage over non-visual approaches. It often leads to faster analysis and better results [1]. Visual representations and interactive techniques take advantage of the human eye behavior, which allows users to see, explore, and understand large amounts of information at once [2].

Many of the available interactive visualization tools are based on the proprietary technology Adobe Flash [3]. But Flash goes against the grain of the open standards by requiring a plug-in to be installed by the user, which is considered a usability and accessibility problem, among other implications [4]. On the other hand, the web standards were born as recommendations proposed by the World Wide Web Consortium (W3C) as an attempt to promote the Internet evolution and to guarantee that all web technologies work well together. For many years, W3C referred to those specifications as recommendations, and this may have contributed to a weak adoption by the web browsers companies. When the Web Standards Project [5] was launched, in 1998, those recommendations were renamed to web standards, and then the support to them became a vital ingredient to any browser or Internet device. The web browsers modernization allows web standards-based rich internet applications to be built, even visualization tools [6].

In order to address those problems, this work proposes the development of a web-based information visualization framework using only web standards and open web technologies. This framework is composed of many interactive visualization modules, each one focused on a certain kind of dataset, and also provides the basic infrastructure that allows its extension by developing new visualization modules. As a result, a fully functional framework with four visualization tools was implemented. These visualizers are completely different since they implement different techniques suitable for different number and type of dimensions. The bubble chart module supports up to four dimensions, the great circles map represents relationships between geographic-related data, the Choropleth map allows a variable to be compared across regions on a map, and finally, the parallel bars chart, an implementation of a new technique proposed by this work, allows *n* dimensions to be represented in a bi-dimensional space. The following sections will present related works, the reference model, the framework and a case study.

II. RELATED WORKS

Perhaps the first framework for information visualization was the Information Visualizer [7]. After that, the popular Microsoft Excel also could generate charts based on a spreadsheet, but beyond running on desktop, it's a proprietary software. Regarding end user visualization applications, Tableau [8] and Spotfire [9] are desktop applications which provide advanced information visualization tools to analyze large amounts of data.

Many other tools were developed in the recent years to allow developers build their own visualizations. Among the ones written in Java, Processing [10] is a famous one. It's a visual programming language and environment for people who want to create images, animations and interactions, but works on a lower-level since it requires some programming skills. Besides this, there are many possibilities to build complex

visualizations, but its target is the desktop environment as well. On the other hand, the target of Prefuse [11] is the web environment, and it defines itself as a toolkit for interactive visualizations creation. The Prefuse framework provides both Java and Flash tools, is freely licensed, but its strength is tree and graph structures, besides also requiring coding skills. More friendly, ManyEyes [12], also written in Java, is a website which provides a service that allow users to upload datasets, create visualizations, publish and comment, but cannot be downloaded and used as a product, just as a public service, which is problem for users that just want to visualize their data but don't want to share them.

Fusion Charts [13] is also another Flash tool for interactive visualization, which is largely used worldwide. It includes lots of visualization techniques, most of them traditional and simple techniques such as pie, bar and line charts. Besides not including complex visualizations, it's a paid product. Part of its code is written in JavaScript and HTML 5.

Among the JavaScript approaches, most of them require coding skills. Raphael [14] is a library that makes it easy to create cross-browser SVG [15]. ProcessingJS [16] is a JavaScript port of Java Processing, so it works as a visual programming language, in a similar way as the Java tool does. The solutions for charts creation are many, but usually include only simple charts and most of them can't be extended.

It's clear that many of the visualization tools were focused on the desktop environment and then started to migrate to the web environment on the last years. At the same time, huge amounts of data became available to the public on the Internet, some of them also accessible through webservices. OPENEDEYES architecture is ready to handle this new scenario, since the data server is independent from the application server. So, data can come from any place (regardless they are on the Internet or stored in a local file), it just needs to be formatted according to what OPENEDEYES expects.

As exposed above, the existing tools can be divided into two main groups. In one group, web applications that implement simple techniques or require advanced knowledge. In the other group, desktop applications that implement more advanced techniques. Both groups could also be divided into free software and non free software. In this context, OPENEDEYES would be placed in an intersection between those two groups since it is web based, open source (available at [17]) and implements advanced techniques without requiring advanced knowledge. This way, many important features can be found in one single solution. It's written in JavaScript and based on web standards, avoiding proprietary technologies such as Adobe Flash. It runs on web browsers, making it naturally available for all platforms. Works on both lower level and high level, so users who have programming skills can extend it by developing new visualizers, while novice users can just generate a visualization for some dataset by using one of the available visualizers. It's fully available for download and licensed as free software, so anyone can download it, modify it and use it. Advanced techniques are available, such as a new technique introduced by this work called *parallel bars*. Conceptually, OPENEDEYES was based on the reference model to be presented in the following section.

III. Reference model

The information visualization process includes some components that are common to many techniques, regardless the data domain. These components are present in reference models [18], [19], [20], which serve as a conceptual framework for structuring infovis applications [11]. This work is based on the reference model shown in Figure 1 [21], modified to show how each component is mapped to an open web technology.

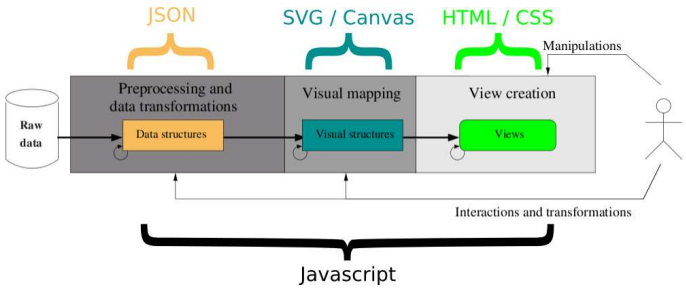

Fig. 1. Adapted reference model for this work based on [21]

The pipeline starts with the raw data, extracted from databases or sensors, for example. Pre-processing turns raw data into a normalized, JSON [22] structure. This first step is out of scope of OPENEDEYES, it expects as input a well formatted JSON structure. These JSON data tables can be static (stored in a file system) or dynamically returned by a data server that provides a webservice. An advantage of this approach is that the same data can be used by different web applications. Data will be graphically represented as visual structures, by a process called visual mapping. Visual structures in OPENEDEYES are SVG elements (vectors), or a Canvas bitmap (HTML 5 element) [23]. Finally the views are created, which are HTML documents [23] formatted using CSS sheets [24]. The transition between each stage of this pipeline is done by pure JavaScript, as well as the user interactions after the view creation. These interactions may or may not perform new requests to the server.

IV. The framework

Regarding its extensibility, OPENEDEYES can behave as a white box framework or as a black box framework. White box frameworks, or architecture-driven, are those that extensibility is achieved by adding new classes. These classes should be included in the framework source code, so it requires some level of knowledge about the framework internals and also programming skills. On the other hand, black box frameworks, or data-driven, are those that don't require much knowledge about the framework internals, since extensibility is achieved by defining configuration parameters [25]. So, it doesn't need much deep knowledge. OPENEDEYES behaves as a white box framework when a developer creates a new visualizer for it and it behaves as a black box framework when a user creates a new view using one of the existing visualizers. The execution flow for it is presented in Figure 2.

The developer - responsible for creating the visualization - includes OPENEDEYES in his application (by just actually including a single installation script) and defines the

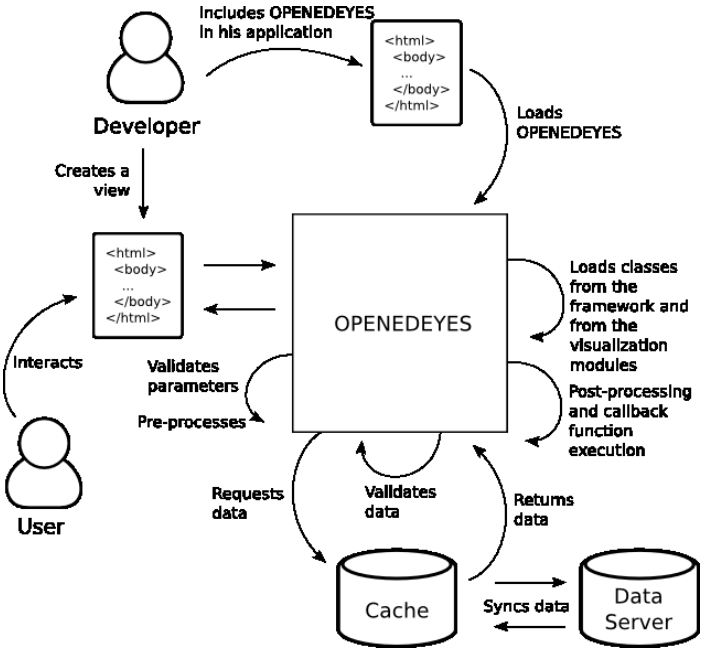

Fig. 2. Global execution flow of OPENEDEYES

configuration parameters and interface controls. The user - responsible for understanding the dataset represented on the visualization - uses those controls to interact with the view generated by OPENEDEYES. The framework is responsible for handling the current state of visualization, current values of configuration parameters, data request and data preprocessing. Interface controls added by the developer communicate with this visualizer by messages, which can be achieved by calling public methods provided by OPENEDEYES. These methods allow changing configuration parameters, setting which variable is represented on each dimension, enabling and disabling dimensions, filtering the dataset, zooming in and out, and so forth.

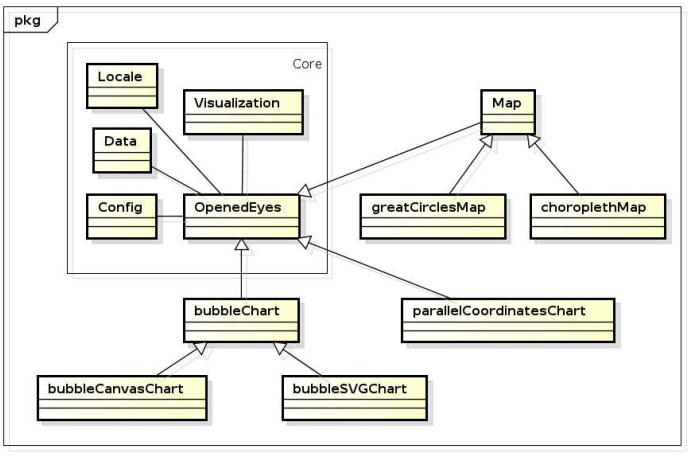

Fig. 3. Simplified OPENEDEYES class diagram

A simplified OPENEDEYES class diagram is represented in Figure 3 (methods are hidden). Five classes (wrapped in the rectangle in Figure 3) compose the framework core. The other classes represent each visualization module, but they are just the visualization modules already implemented in this work as an attempt to show the framework extensibility. A new visualization module could be implemented by creating a new class that inherits from the OpenedEyes class, or that inherits from any other visualization class. The core classes are:

- Data: Handles input data. As mentioned previously, the data table for OPENEDEYES is a JSON structure, that can be returned by AJAX [26] or by JSONP [27], regardless it's a local resource or a remote one. The choice for JSON instead of another popular format, such as XML [28], is because JSON is light-weight (faster and cheaper to be transfered through a network), easy to be read or written (can be manipulated in lots of programming languages, making it easy to be sent by a data server), subset of JavaScript (language on which OPENEDEYES is written), among other reasons [29]. Tests also suggest that JSON requires less resources and is faster than XML [30]. XML has the advantage of being a format for universal representation of data, simple, human-readable and self-explained [30], advantages not so important as the JSON ones for the objectives of this work. Section VI-F1 describes the format of the input data table.
- Visualization: Defines helper methods for visualization. Visual structures can be elements from an SVG document or part of a bitmap drawn on a Canvas element. The main difference between those two approaches is that SVG, as being actually XML, represents each scene element independently as an XML element, which allows this element to be controlled independently of the rest of the image, which usually results in smoother transactions between two states of the visualization. On the other hand, when many visual structures are present, the SVG approach can lead to performance failures, because many elements would have to be stored in memory, and so in this case the Canvas approach would be a better choice. Since it's a single bitmap, it can represent complex visualizations, although it does not allow each scene element to be controlled independently, what means that any change on the visualization requires a full regeneration of the scene.
- Config: Handles configuration parameters. This module is responsible for defining each configuration parameter, validation, default values, current state and visualization update in case of changing any configuration value.
- Locale: Handles translations. This class allows visualization to be available into different languages.
- OpenedEyes: Main class of the framework. It's actually an abstract class that defines the basic structure that any visualizer must have. All visualizers implemented on OPENEDEYES must inherit from this class (or from any other class that descends from it).

The visualization modules are interactive and implement non-trivial techniques. Multiple inheritance and abstract

classes are used to allow extension of the framework. For example, classes OpenedEyes, bubbleChart and Map are abstract classes, but Map and bubbleChart inherit from OpenedEyes. The Map class defines tasks that are common to many geographic visualizations, while bubbleChart calculates values for visual structures allowing its child classes to represent them in different ways. Another visualization module implemented is a variation of the parallel coordinates chart [31], proposed by this work, that consists in replacing each line by a ”polygonal line” which thickness is directly proportional to the represented value. Each polygonal line represents a subset of the input dataset instead of a single element. The following section will show a case study that was made using those visualizers along with real data.

V. Case study

As a case study for this work, it was chosen a dataset from SaferNet Brazil [32], which stores data about reports of crimes against the human rights on the Internet. The choice considered mainly the fact that this database had different data types, such as multidimensional, geographic and temporal, that are related in non-trivial ways. Then, a lot of hidden information could be revealed by using information visualization tools.

Figure 4 represents a simplified diagram of the data model. Only the used entities, relationships and attributes are represented, since the original database is much bigger and more complex.

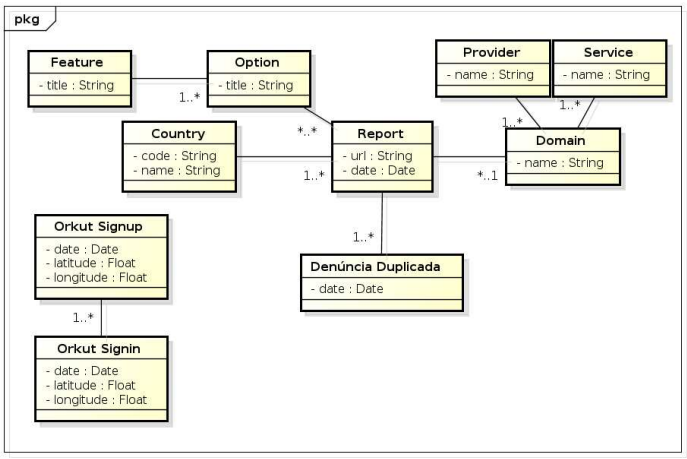

Fig. 4. Database used in the case study

The report is the most important entity in the data model and represents each reported URL. Each report is associated to an Internet domain, which can be related to a provider and a service. A report has many qualifications, defined through feature options. The country where the reported website is hosted is stored. If some URL is reported twice, a new report record won’t be created, instead, a duplicate report is created associated to the first one. Besides these, it was used also some geo-referenced data about an operation of Federal Police of Brazil that fought child pornography in the Orkut social network.

This dataset carried some questions. What providers were responsible for hosting the most reported content? And the ones that most removed criminal content when notified? To analyze many aspects of a report, some multidimensional visualization would be necessary. For the geo-referenced data, a geographic visualization would be probably the most intuitive way, both by region and exact location. Clarifying these questions, and following the behaviour through time (since most of this data is temporal), would help to understand the evolution of the reports and possibly predict some trends in cyber crime.

Four visualization modules were developed and each of them got as input a datafile from the above dataset. The respective results are presented in the following section.

VI. Results

The following subsections explain each developed visualizer and how they were used to gather information from the input dataset. This section also explains how to develop a new visualization module to the framework and shows the advantages of having multiple visualizers that use the same format for input data.

A. Bubble Chart: Up to four dimensions

The Bubble Chart module implements a variation of the scatter plot, where data points are replaced by bubbles that define a new dimension. In a bubble chart, each element is represented by a bubble which area is proportional to some variable from the dataset. In this implementation, each bubble has four visual properties (color, size, horizontal position and vertical position), so this chart may have from one to four dimensions (since any dimension can be disabled at any time). There are two classes,the first one uses SVG and so offers a richer experience as any modification in the chart is reflected in an animated way. The other one uses Canvas and so might be a better choice when the input dataset has lots of elements, although the transition is not so smooth.

Each OPENEDEYES visualization module includes a default interaction interface (black box behavior), although new controls can be added (white box behavior). This default interface has four select menus where each of them defines which variable will be represented in each dimension. It’s also possible to filter and zoom the dataset by clicking on some bubble, that will be removed and all the chart will be redrawn as if the removed element didn’t exist in the dataset (Figure 5).

On the example on Figure 5, it’s possible to verify which are the providers responsible for hosting reported content through time. Each element is an Internet service provider, like Google or Yahoo, for example. There are four attributes, that are the number of reports (plotted in y-axis), the number of distinct URLs (plotted in x-axis), the number of URLs that are still online (represented by the bubble size), and the provider name (which defines the bubble color). There is also a timeline divided by years. Figure 5 shows that in 2011, excluding reports related to Google services, Twitter (the light blue bubble) and Facebook (the dark blue bubble) were two services related to many reports, reflecting the increasing usage of both networks in Brazil.

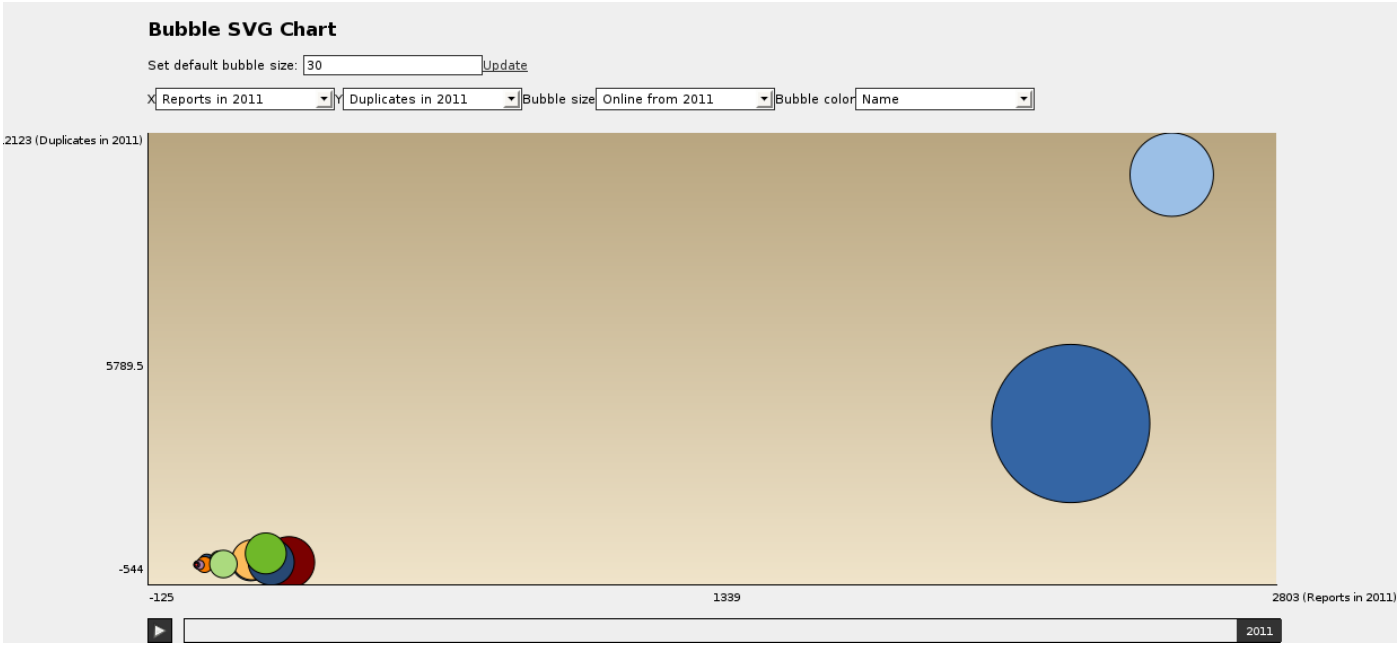

Fig. 5. Reports by provider in 2011, ignoring Google

B. Parallel Bars Chart: Representing n dimensions in a two-dimensional space

Parallel coordinates chart can represent, in a bi-dimensional space, *n* distinct dimensions of dependent attributes. Each attribute is represented by a bar, which are equally spaced and positioned parallel [31]. Parallel coordinates chart represents each element from the dataset as a line that crosses each attribute bar. A problem with this visualization is that it can lead to poor performance when many elements are present.

A variation of this technique, called *parallel bars* (Figure 6), is proposed in this work. Instead of representing each element as a line, each subset of elements is represented by a polygonal line, which thickness is proportional to the value at that bar, so the subset is filtered as the line crosses the bars. This way less lines are represented without information loss, which leads to better performance. Global analysis is possible since each line groups elements with the same features.

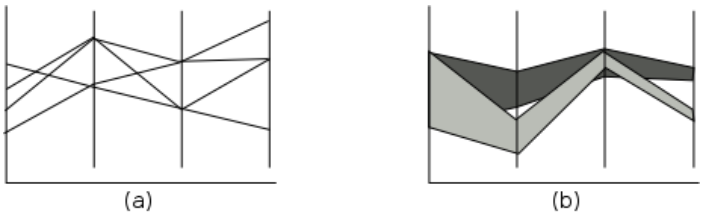

Fig. 6. Main difference between the way that data is represented in (a) parallel coordinates and (b) parallel bars

In order to be able to compare the proposed technique with the classic parallel coordinates one, the latter was also implemented in this same visualization, as a configuration option. When instantiating the visualization, the user can set the option *group*, which has a boolean value. When *true*, rows from the input dataset will be grouped by value, resulting in polygonal lines (i.e., parallel bars technique will be used). On the other hand, when the *group* value is *false*, the classical parallel coordinates technique will be used, so, each row from the data table will be rendered as a single line in the chart. Figures 7 and 8 show this difference more clearly, since one of them implements the parallel bars technique, the other implements the parallel coordinates technique and both of them use exactly the same dataset and parameters. On Figure 7, it's possible to see three subsets of the original dataset: racism content hosted in Brazilian servers (represented by the orange line), Nazism content hosted in Germany (represented by the green line) and child pornography content hosted in The Netherlands (represented by the blue line). As the polygonal line becomes thiner as it crosses the bars, it's easier to compare data between the bars and between the lines. Also, it's fast to render because just a single polygon is rendered for each line. On Figure 8, each row from the data table is represented as a single line. Rows with same values gain the same color, for easier analysis. But still, it's not so intuitive, because all lines have same width, and not so fast, because for each row from the dataset, a line will be rendered on the chart.

The main difference between this visualizer and the other ones developed in this work is that this one requires the definition of not only the variables that will be plotted in each dimension, but also which are these dimensions. Each dimension is represented by a bar that is related to some attribute from the dataset. The default interface allows bars to be added, removed and reordered. It also allows new polygons to be added, removed or manipulated, by just clicking on the

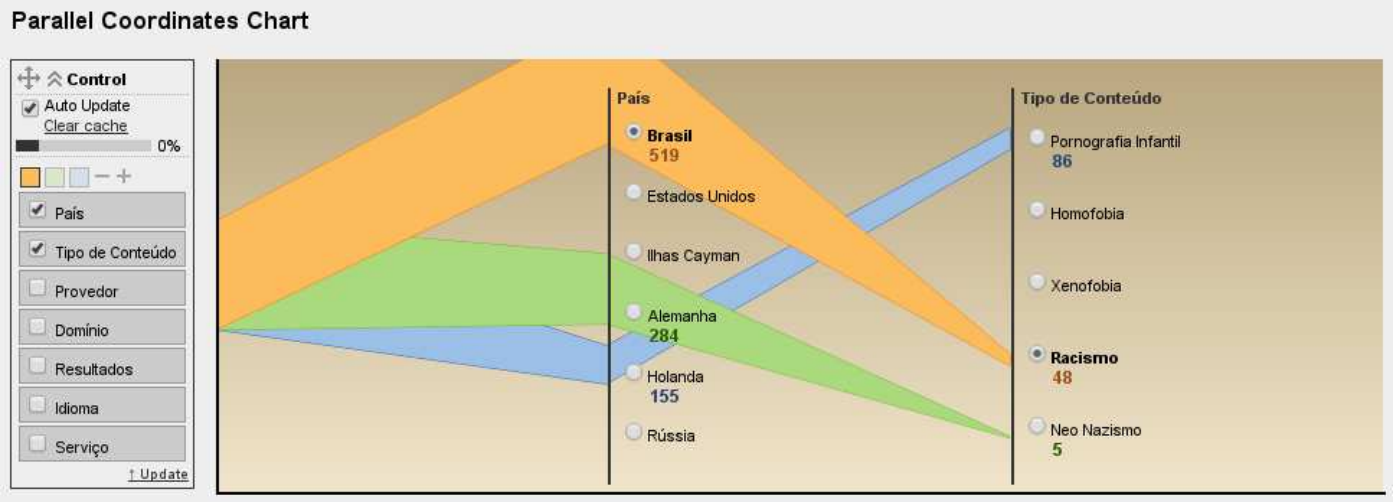

Fig. 7. Parallel bars: dynamic-width polygonal lines for each subset of the input dataset

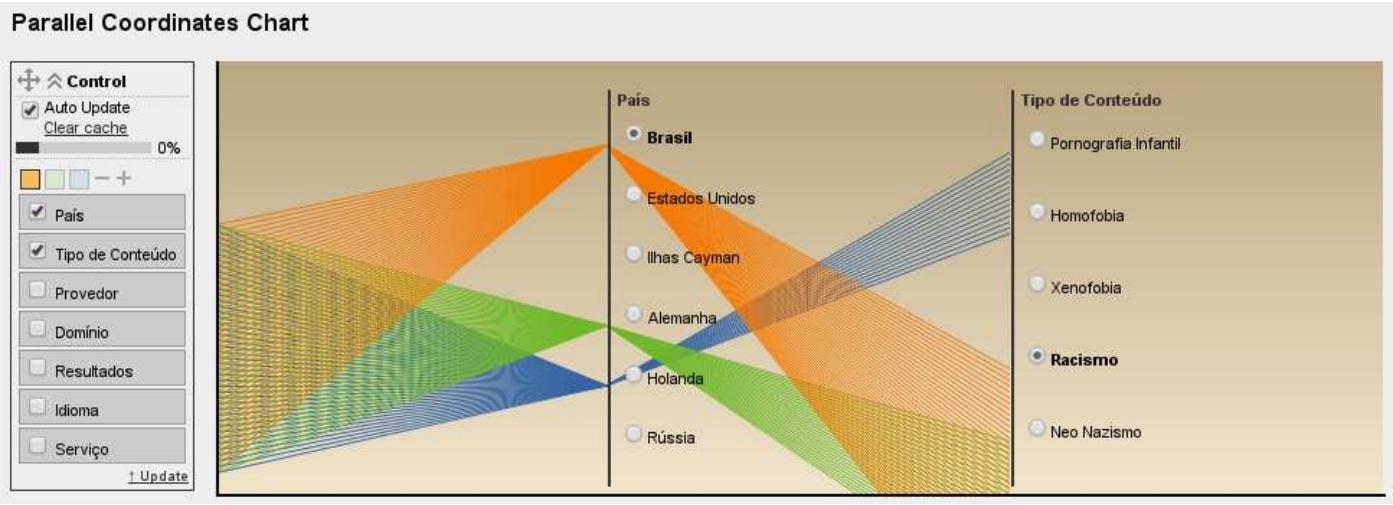

Fig. 8. Parallel coordinates: fixed-width lines for each element of the input dataset

value for some attribute. Another big difference is that Parallel Bars usually will demand new data requests. A local cache, implemented using local storage [33], is available to avoid unnecessary requests. The cache can be available even after the browser session is over, making it useful for datasets that won't change much over time. In case of change, a "clear cache" feature is also available.

The dataset picked to be visualized using parallel bars is formed by reports and their attributes. Figure 9 shows a parallel bars view that has seven attributes, from which three are active (the first three checked ones in the left control box). The order of the attributes in the control box is reflected in the bars order in the chart. The data shown in the chart is related to 2008, as marked in the timeline. There are four profiles of data, each one represented by a colored polygon, from which the blue one is active, what means that any interaction made directed in the chart will reflect only in this line, while interactions done through the control box will reflect in all lines.

A big dataset can be filtered as the bars are being crossed. For example, from Figure 9 it's possible to see that UOL was the Brazilian Internet service provider with most reports of cyber crimes, from which almost half of them was related to child pornography content. This view also shows the number of child pornography reports on the Internet related to other three Brazilian providers in 2008.

Another interesting aspect of this chart is that it implements one of the most important features that a visualization tool must have: global view first, and zoom and filtering on demand. The chart starts with a global view of the whole dataset, and as new dimensions are added (on demand) and as a polygonal lines crosses those bars, the dataset is being filtered by the select value on each bar (filtering), and details become more evident (zooming). In a nutshell, the leftmost side of the chart will provide a global view of the dataset, while the rightmost

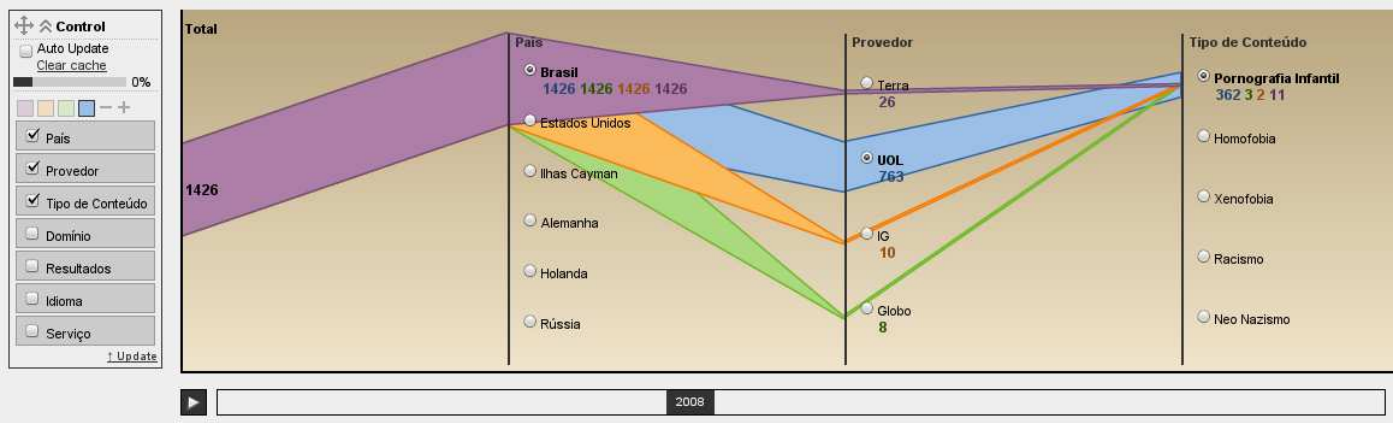

Fig. 9. Brazilian providers that hosted child pornography reported content in 2008

side of the chart will provide a detailed view of a subset of this dataset.

C. Choropleth Map: Shading areas in a thematic map in proportion to the measurement of a variable

Choropleth map is a conventional way to plot geographic density data [34]. This technique uses a sequence of color tones or shadows to apply over the regions. Areas with higher values get darker colors, while areas with lower values get lighter colors. In this OPENEDEYES implementation, the map regions are colored according to value of the represented numeric variable. The colored regions define a dimension of this visualization. The other dimension available is defined by bubbles over regions, which areas are proportional to the represented variable. This alternative is present because in some cases colored maps are not the best way to visualize data, since bigger regions get more attention than the smaller ones, which can lead to misinterpretation [35].

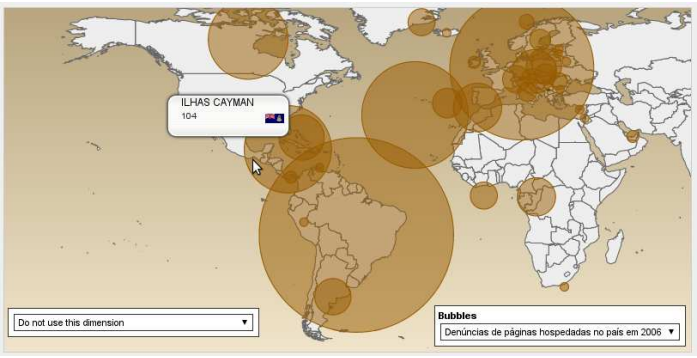

Fig. 11. Cayman Islands were responsible for hosting many websites of child pornography content in 2006

The example on Figure 10 shows the number of reported websites hosted by countries. The United States host most of them, so it was removed from the dataset to allow other countries to be analyzed. As shown in Figure 10, in 2007 Brazil was responsible for hosting a lot of reported pages, followed by Australia. The Netherlands and the Cayman Islands are two countries that deserve attention, but using only colors is not possible to notice that, since their regions are small. This is a case where using bubbles instead of colors is a better approach, like is shown in Figure 11.

D. Great Circles Map: Connecting points on the surface of Earth

A great circle is a section of a sphere that contains the diameter of that sphere. The shortest path between two points in a sphere is a segment of a great circle [36], not a straight line. As the Earth is not a perfect sphere, the shortest distance between two points is not exactly a great circle, but in this work an approximation is made and so the Earth is considered a perfect sphere, as the goal is not to measure distances, but to identify relationships between points, which can be clearly accomplished with this approach.

The following equations are used in order to convert coordinates to pixels. The variables *lat* and *lon* are the coordinates of the point. Parameters *w* and *h* are denoted in pixels and represent the dimensions of the chart. Finnaly, *tx* and *ty* are related to transformations that can be applied to the map (horizontal and vertical, respectively), which is identified by looking at the SVG viewport.

$$x = \frac{w(lon + 180)}{360} - tx$$

$$y = h - \frac{h(lat + 90)}{180} - ty$$

The opposite can be also accomplished by using the following equations, but *x* and *y* are input parameters while *lat* and *lon* are output:

$$lat = -\left(\frac{(y - h) * 270}{h} + 90\right)$$

$$lon = \frac{x * 360}{w} - 180$$

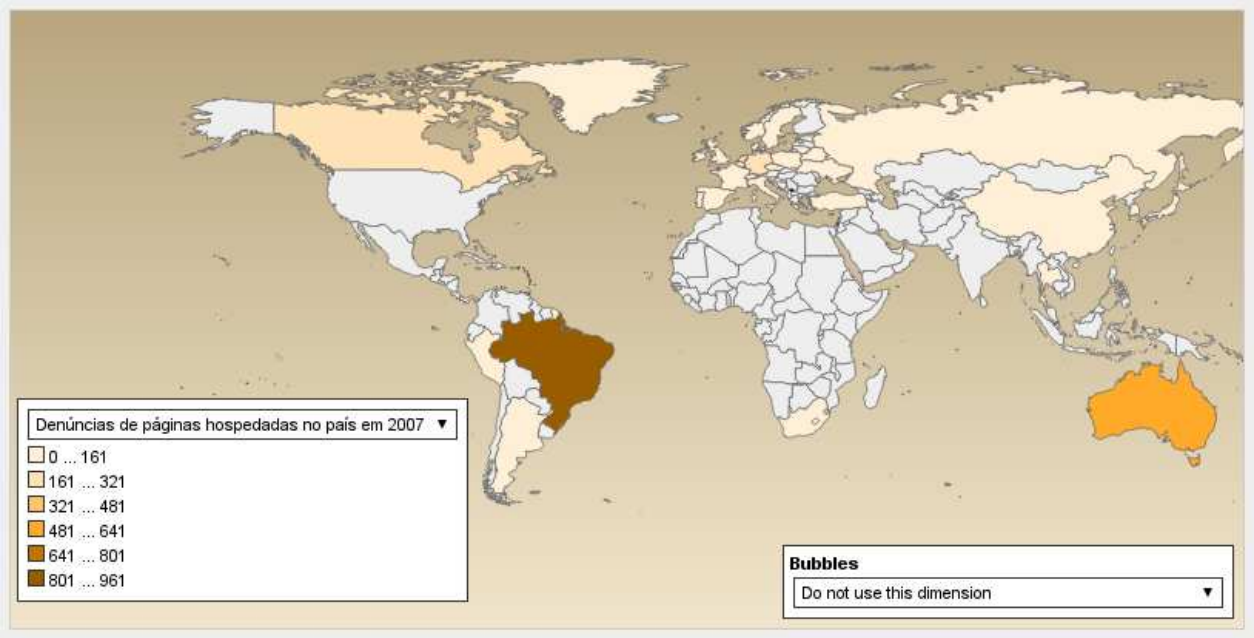

Fig. 10. Countries that host most of reported content in 2007, excluding the United States

The example on Figure 12 shows how the Orkut child pornography offenders act. A Brazilian Federal Police operation called Turko was launched in May 2009 to search and seize computers in 103 different locations in Brazil. Data related to this operation was used to generate the map on Figure 12. The red points show the IP addresses used by the offender to create an Orkut profile to spread child sexual abuse material, and the yellow ones show the IP addresses used by the offender to access his illegal profile in Orkut. There are lines connecting the location where the profile was created to the location from where the offender accessed his account. This view shows that offenders and the victims are both located in Brazil (and also in India), despite the fact that the content was hosted in Google servers in the United States. This visualization shows that a unique metric or point-of-view usually is not enough to get the big picture of a situation.

E. Comparison between the visualizers

As mentioned previously, all visualizers implemented on the framework recognize the same input format. This way, the same datafile can be visualized in different ways, leading to different information and interpretation about the dataset. Consider data about reported content hosted in countries by some Internet service providers. This datafile used in this comparison has three non-numeric variables (provider, country and content type) and two numeric variables (distinct reported URLs and total number of reports), and are related to time. This datafile will be represented in three different visualizations: bubble chart, parallel bars chart and Choropleth map. By default, all visualizers have a timeline, so it's possible to filter data by time in all three visualizations, the same way.

As Figure 13 shows, parallel bars chart has the advantage of representing many different non-numeric variables, each one as a bar, but just one numeric variable, which is filtered by crossing the bars. There is geographic data associated to the dataset, but on the visualization it has no special representation, it's just a bar as any other dimension. This way, it's possible to analyze very detailed and deep information, for example, UOL was a provider responsible for hosting 25% of racism content hosted in Brazil in 2006.

On the other hand, a map is a much more intuitive way to analyze geographic data, as illustrated on Figure 14. Each region is colored considering a numeric variable from the dataset. But here there are only two variables being represented (total reports and country), while on the parallel bars chart four variables were being represented. On this example, it's possible to see that Brazil, followed by Australia and Germany, were countries responsible for hosting many reported contents on 2006 (The United States was removed to facilitate analysis).

The bubble chart is a choice that can handle many numeric variables, since it has three dimensions for that (x-axis position, y-axis position and bubble size), but just one dimension for non-numeric variable (the bubble color). So, for this dataset, the two numeric variables are represented in the axises and just one non-numeric variable can be viewed at a time, as shown on Figure 15, where the hosting provider is expressed as the bubble color. This example shows that on 2006, UOL (represented by the orange bubble) was the provider responsible for hosting many reported contents, both distinct URLs and total reports.

These examples show that having different visualizations with the same input data format is a powerful way to analyze data from different perspectives. Parallel bars can be used to

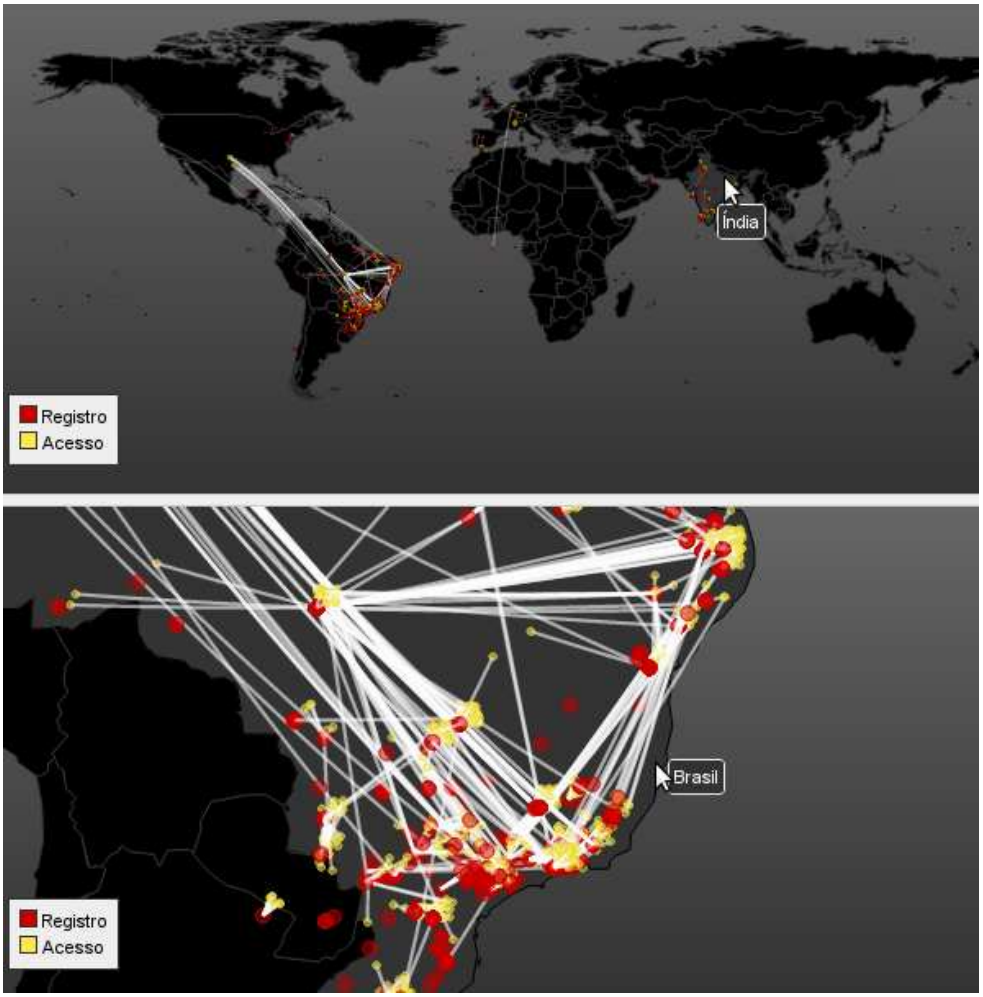

Fig. 12. Great circles used to represent data about Turko operation

analyze most of the non-numeric features of the data, while bubble charts can be used to analyze most of the numeric features of them, and for geographic data, Choropleth maps offer an intuitive and natural way to analyze some variable (numeric or not) about certain region. Also, parallel bars can lead to more detailed and specific information, while Choropleth maps and bubble charts can provide the big picture of some dataset.

F. Extending the framework

As mentioned previously, the framework is a hybrid one, which can behave as black box or white box. This section will explain how it can be extended on these both ways. Reuse is achieved by implementing a new visualization module based on an existing one or by using an existing visualization module to view some dataset.

1) Creating a view: In order to create a view, the developer needs to prepare the data in the way OPENEDEYES understands it. It's enough to provide a JSON structure as a static plain text file or automatically generated by a webservice. Figure 16 compares a typical table structure with the OPENEDEYES JSON format. OPENEDEYES also includes a script Shell that converts a file from CSV to its format.

In case of error, the data server can return a JSON structure with an error message, which will be presented to the user after evaluated by OPENEDEYES. An error message should be formated as represented on Figure 17.

After releasing the input data, the developer just needs to include a single *script* tag in the application, which contains the installation steps. This script, when loaded, will automatically include the other required files and trigger the correct functions in order. Finally, the developer just needs to instantiate a class of OPENEDEYES that represents a visualizer. The code in [17] includes demonstrations for each available visualizer.

2) Implementing a new visualization module: The framework can be extended by implementing a new visualization module. In order to do this, it's necessary to create a new class that inherits from *OpenedEyes* class or any other existing visualization module. The following class attributes must be

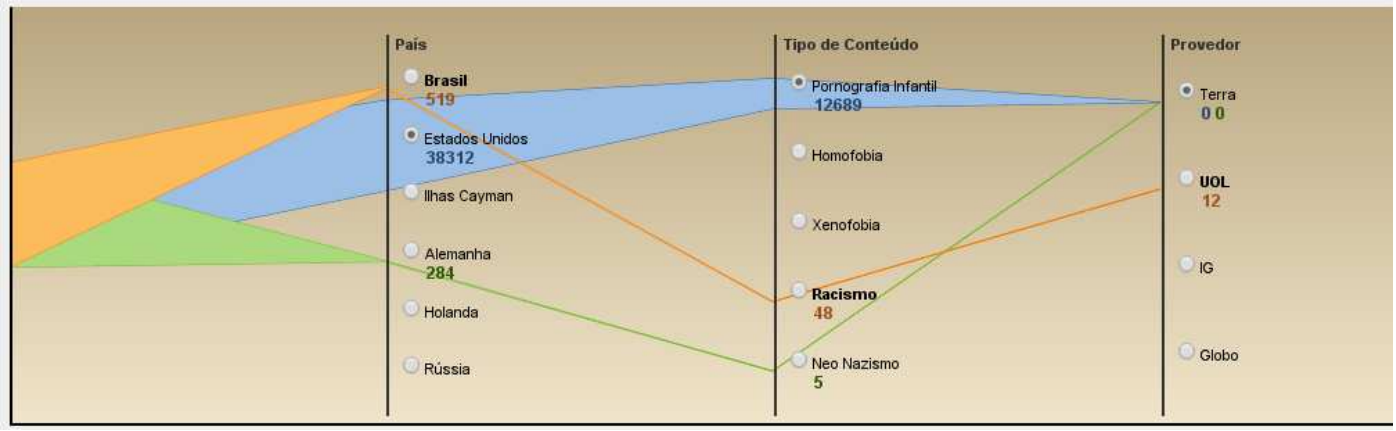

Fig. 13. Common datafile represented in parallel bars: all three non-numeric variables and one numeric variable (total reports) are represented in this chart

Name	Country	Age
John	BR	25
Paul	US	32

```
{
  "header" : [ "Country",
               "Age" ],
  "values" : {
    "John" : {
      "Country" : "BR",
      "Age" : 25
    },
    "Paul" : {
      "Country" : "US",
      "Age" : 32
    }
  }
}
```

Fig. 16. A typical data table and its corresponding OPENEDEYES JSON structure

```
{
  "error" : "Some error message",
}
```

Fig. 17. OPENEDEYES error format

defined:

- prettyName: Readable name of the visualization module.
- name: Unique name for the module.
- dependencies: Additional JavaScript and CSS files that must be loaded when the module is loaded.
- customConfig: Additional configuration parameters for the module with their default values.

Also, the following methods must be implemented:

- createInterface: Builds the default user interface for the visualization.
- postLoad: This method performs the visual mapping on the input data.
- updateVis: Updates the visualization when some parameter (configuration, input, filter or dimensions) is changed.

There are additional methods and attributes that can be overwritten, and new methods and attributes can be added.

The empiric results obtained in this section indicates that the framework is flexible and usable, being capable of building information visualizations with different relationships, as well as implementing various visualization techniques, even new ones proposed by the users.

VII. Conclusion

This work presented OPENEDEYES, a web standards-based framework for information visualization on the web written in JavaScript. Its architecture is modular, defined by five core modules, responsible for common actions and basic structure, and four visualization modules, which implement advanced visualization techniques to fit distinct datasets. As a framework, its hybrid behaviour, acting sometimes as a black box framework or as a white box framework, makes it usable for developers and for users, from novice to advanced. Interaction is such an important ingredient in visualization, and so all developed visualization modules include a default interaction interface but also allow new controls to be added. OPENEDEYES comes to fill a gap in the visualization tools world by offering tools which are, at the same time, web based, interactive, simple but robust, and even innovative, besides providing a structure that allows new visualizers to be added. Future works include studying how this implementation scales for larger datasets, whether there are performance failures, how they can be mitigated, and evaluate the usability of the visualizers for users and developers. Future works could also try to extend OPENEDEYES by generating visualizations of real-time released data.

References

[1] D. A. Keim, “Information visualization and visual data mining,” *IEEE transactions on visualization and computer graphics*, vol. 7, no. 1, 2002.

[2] J. J. Thomas and K. A. Cook, *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr, 2005.

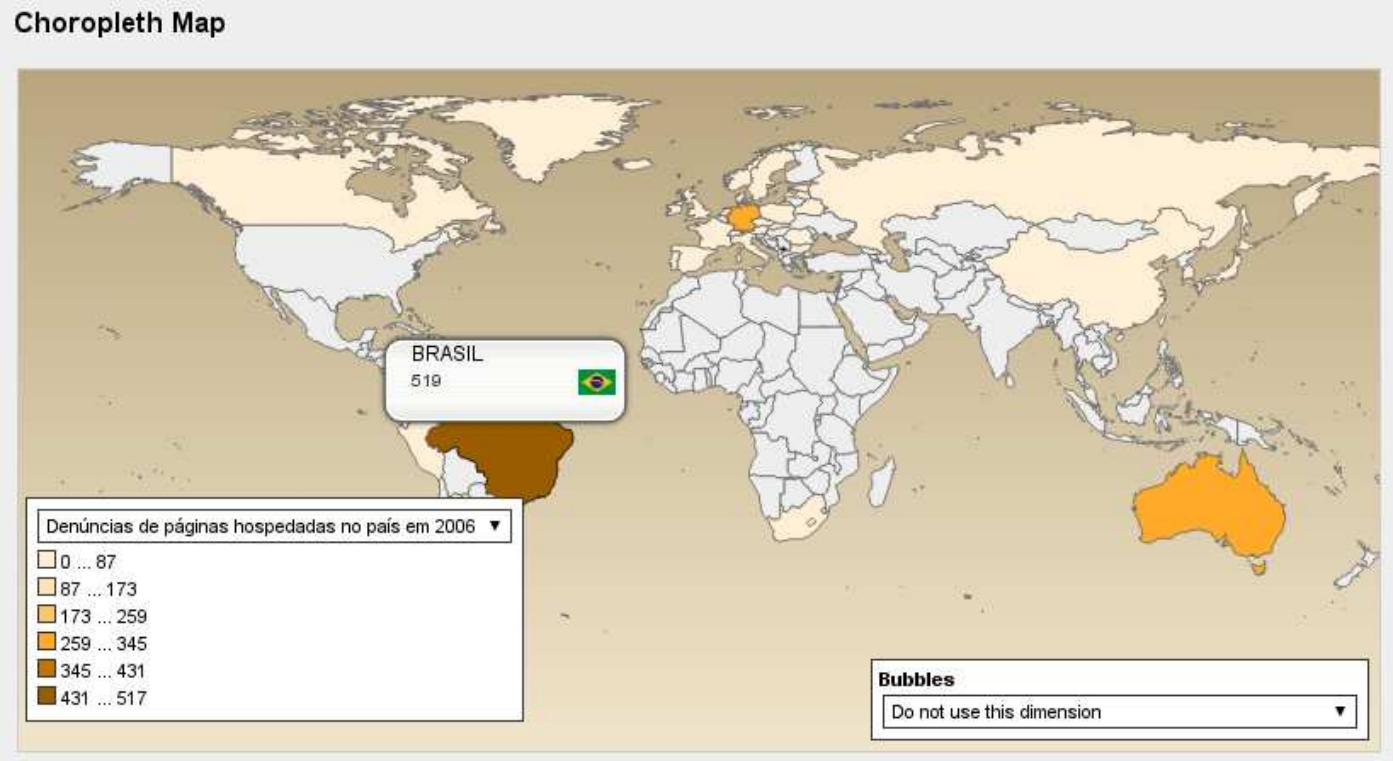

Fig. 14. Common datafile represented as a Choropleth map: just one non-numeric variable (the country itself) and one numeric variable (total reports) are represented in this chart

[3] Adobe, "Flash," 2011, available at: http://www.adobe.com/br/products/flash.html (last accessed December 11, 2011).

[4] J. Nielsen, "Flash: 99% bad," 2000, available at: http://www.useit.com/alertbox/20001029.html (last accessed December 11, 2011).

[5] T. W. S. Project, "About the web standards project," 1998, available at: http://www.webstandards.org/about/ (last accessed December 11, 2011).

[6] R. Kosara, "The state of information visualization, 2011," 2011, available at: http://eagereyes.org/blog/2011/state-of-infovis-2011 (last accessed December 11, 2011).

[7] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer, an information workspace," in *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, ser. CHI '91. New York, NY, USA: ACM, 1991, pp. 181–186.

[8] Tableau, "Tableau," 2012, available at: http://www.tableausoftware.com (last accessed December 11, 2011).

[9] TIBCO, "Spotfire," 2012, available at: http://spotfire.tibco.com/ (last accessed December 11, 2011).

[10] B. Fry and C. Reas, "Processing: A programming handbook for visual designers and artists," 2007.

[11] J. Heer, S. K. Card, and J. A. Landay, "Prefuse: a toolkit for interactive information visualization," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '05. New York, NY, USA: ACM, 2005, pp. 421–430.

[12] F. B. Viegas, M. Wattenberg, F. van Ham, J. Kriss, and M. McKeon, "Many Eyes: a Site for Visualization at Internet Scale," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1121–1128, 2007.

[13] FusionCharts, "FusionCharts," 2011, available at: http://www.fusioncharts.com/ (last accessed December 11, 2011).

[14] D. Baranovskiy, "Raphael," 2011, available at: http://raphaeljs.com/ (last accessed December 11, 2011).

[15] W3C, "Scalable Vector Graphics," 2011, available at: http://www.w3.org/Graphics/SVG/ (last accessed December 11, 2011).

[16] J. Resig, "ProcessingJS," 2011, available at: http://processingjs.org/ (last accessed December 11, 2011).

[17] C. S. B. Almeida, "Openedeyes public git repository," 2013, available at: https://github.com/caiosba/openedeyes (last accessed August 01, 2013).

[18] S. Card, J. Mackinlay, and B. Shneiderman, *Readings in information visualization: using vision to think*, ser. The Morgan Kaufmann series in interactive technologies. Morgan Kaufmann Publishers, 1999.

[19] E. H. Chi, "A taxonomy of visualization techniques using the data state reference model," 2000.

[20] R. B. Haber and D. A. Mcnabb, "Visualization idioms: a conceptual model for scientific visualization systems," *Visualization in Scientific Computing*, 1990.

[21] R. Mazza, *Introduction to information visualization*. Springer, 2009.

[22] D. Crockford, "RFC 4627 - the application/json media type for javascript object notation (JSON)," 2006, available at: https://tools.ietf.org/html/rfc4627 (last accessed December 11, 2011).

[23] W3C, "HTML 5," 2011, available at: http://www.w3.org/TR/html5/ (last accessed December 11, 2011).

[24] ——, "Cascading style sheets," 2013, available at: http://www.w3.org/Style/CSS/ (last accessed August 01, 2013).

[25] M. E. Markiewicz and C. J. P. de Lucena, "Object oriented framework development," *Crossroads*, vol. 7, pp. 3–9, July 2001.

Fig. 15. Common datafile represented in a bubble chart: just one non-numeric variable (the hosting provider) and two numeric variables (total reports and distinct URLs) are represented in this chart

[26] J. J. Garret, "AJAX: A new approach to web applications," 2005, available at: http://www.adaptivepath.com/publications/ essays/archives/000385.php (last accessed December 11, 2011).

[27] B. Ippolito, "Remote JSON - JSONP," 2005, available at: http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/ (last accessed December 11, 2011).

[28] W3C, "Extensible Markup Language," 2011, available at: http://www.w3.org/standards/xml/ (last accessed December 11, 2011).

[29] D. Crockford, "JSON: The fat-free alternative to XML," 2006, available at: http://www.json.org/fatfree.html (last accessed December 11, 2011).

[30] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta, "Comparison of JSON and XML data interchange formats: A case study," *Scenario*, vol. 59715, pp. 157–162, 2009.

[31] A. Inselberg and B. Dimsdale, "Parallel coordinates: a tool for visualizing multi-dimensional geometry," in *Proceedings of the 1st conference on Visualization '90*, ser. VIS '90. Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, pp. 361–378.

[32] S. Brasil, "Quem somos," 2008, available at: http://www.safernet.org.br/site/institucional (last accessed December 11, 2011).

[33] I. Hickson, "Web storage," 2011, available at: http://dev.w3.org/html5/webstorage/ (last accessed December 11, 2011).

[34] D. Cuff and M. Mattson, *Thematic maps: their design and production*, ser. University paperbacks. Methuen, 1982.

[35] IBM, "Many Eyes: Maps," 2007, available at: http://www-958.ibm.com/software/data/cognos/manyeyes/page /Maps.html (last accessed December 11, 2011).

[36] E. Weisstein, "Great circle," 2011, available at: http://mathworld.wolfram.com/GreatCircle.html (last accessed December 11, 2011).