Multiple coordinated views of massive geo data using tree maps and choropleth maps

Robert Schäfer
Department of Computer Graphics, Hasso-Plattner-Institut

October 2, 2017

Contents

Abstract

Numerous visualization techniques exist for data-driven decision support systems. Two of which, namely tree maps and choropleth maps are especially useful when dealing with hierarchical and geographical data respectively. While a choropleth map is an sensible choice for geographical data, there is no obvious mapping of arbitrary hierarchical data. On the other hand, tree maps allow to visualize arbitrary, hierarchical and multidimensional data using nested rectangles. Although a tree map looks similar to a geographical map, the result may not be as intuitive and comprehensible. There is no predefined layouting for arbitrary data, let alone no well known layouting that everybody feels comfortable with. In this thesis, we propose and evaluate a coordinated multiple view to combine advantages of both visualizations. In one view, the user gets an easy access with a familiar geographical map, while in the other view the same data is displayed in a tree map.

Should we mention the combination of multiple data sets here?

1 Introduction

The human brain processes visual information better than it processes text. As a result, the most tangible data analyses usually come with some sort of data visualization. Data visualizations on a computer allow for user interactions and different levels of granularity according to the customer demands, which provides a great user experience. There are a multitude of different techniques. Coordinated multiple views take data visualizations to the next level by exploiting the respective advantages of the used visualizations as much as possible. This combination may yield a greater value to the user, but it is unclear what kind visualization techniques work best together and for which kind of data. How a user interact with coordinated multiple views and how it differs from the use of single visualizations is another question worth to investigate. In this thesis, we consider the question how a combination of a geographical visualization with a hierarchical visualization performs, for what kind of data this combination is suitable and what interaction patterns apply.

Establish the niche, why is there further research on your topic?

Introduce the current research, what's the hypothesis, the research question?

1.1 Motivation

We create data visualizations of multi-dimensional, hierarchical and geographical data. Namely, we develop RUNDFUNK MITBESTIMMEN which is an application for German citizen to publish which public broadcasts should benefit from their broadcasting fees. The output of this application is a public user ranking and it can be used by broadcasting corporations to evaluate their program. Data visualizations guide media researchers, journalists and the general audience to draw conclusions. In this particular use case, the selection and interaction with the data may happen geographically, but the desired visualization could show e.g. changes over time or relationships within the data.

To explain this a little deeper: Public broadcasting in Germany is organized federally, a German home belongs to the jurisdiction of a public broadcasting corporation. But the produced content can be used all German citizen and it is even required to be free and available to everyone. So this means that e.g. a media researcher might want to select all users from within a certain region, but is actually interested into relationships of broadcasts that are preferred by people from that area.

So the interaction and selection of data should happen in another view than the actual data visualization. Since we deal with geographical data, we use geometry on a map for selection and interaction. For the visualization, we use a different technique, e.g. a tree map if we deal with hierarchical data.

More scenarios here

1.2 Problem statement

2.5D tree maps visualize hierarchical data on a two dimensional canvas and are particularly suitable if the proportions of the data should be emphasized. When dealing with both multidimensional and geographical data, problems arise when other features than geographical features are used for the layouting of the 2.5D tree map. As the order and placement of items depend on their specific values and hierarchy, items that should belong together according to their geographic circumstances may be scattered across the 2.5D tree map. This obstructs the comprehensibility of map and makes it especially difficult to select geographical units of items.

1.3 Research questions

When dealing with multi-dimensional data, is it helpful to have multiple views for these dimensions? What are best practices for the implementation of coordinated multiple views? Is it great user experience to have one view for interaction and another view for the data visualization to create insights? Do people prefer one view for the interaction, e.g. the geographical dimension, or do they use all views for interaction and visualization alternately?

1.3.1 Hypotheses

Linkage and coupling of coordinated multiple views can be enabled through visualizations, navigations and interaction techniques. The exploration and analysis of multi-dimensional and geographical data can be supported with these coordinated multiple views.

1.4 Objectives

No section without text

Basic coordinated multiple view The existing VISUAL ANALYTICS PLATFORM is supplemented with a basic coordinated multiple view system. Modules, interfaces and functionalities of the coordinated multiple view system are designed, coordinated and prototypically implemented. In particular, a method for arranging multiple coordinated multiple view widgets in a coordinated multiple view layout and storing coordinated multiple view layouts is developed.

Interaction with the coordinated multiple view We develop coupling mechanisms and interaction mechanisms between 2D maps (for map-based representation) and 2.5D tree maps (for abstract information representation). The functionality includes zoom per object or selection with a bounding box. This creates a powerful selection mechanism, which can be used to select the data from the map-based representation in the 2.5D tree map.

Demonstration and evaluation Coordinated multiple view layouts and suitable views are implemented and tested for the selected test data. Based on the test data sets, the coordinated multiple view implementation is examined and evaluated for design criteria[24], general usability aspects[20] and usage for typical visual analytics tasks.

How elaborate is it to conduct a study for the usability of the coordinated multiple view? Who has done that before?

1.5 Methodological approach

A literature research is used to gather knowledge about the state of the art with respect to coordinated multiple views. Existing concepts and implementations available on the internet are examined and reused if possible. Interviews of people from the target group are conducted and the define the requirements for the application. A minimal viable product is developed to further validate the user requirements. Also, common user behaviour is observed during user tests. The prototype is continuously developed to allow for further experimentation.

Scenarios While implementing more features, we have two scenarios with different kind of data and different user requirements:

- In our work on the project RUNDFUNK MITBESTIMMEN we design and prototype visualizations for media researchers. We fully control the database and the database schema as well as on the user facing application on top of it. User requirements are tied to journalists, media researchers, broadcasting corporations. The data has geographical, hierarchical, temporal and correlated characteristics.
- The VISUAL ANALYTICS PLATFORM for administrative data is used as a more general purpose application. There is no single database schema but combatibility with many sources or services. User requirements are potentially unknown and part of the resarch. The usage focuses especially on geographical and hierarchical data.

For the scope of this master thesis we therefore compare implemented visualization and views. How do both approaches differ in development speed, value for the customer? What considerations need to be done regarding the database schema?

What is the area(s) of research, in which this thesis can be placed into?

1.6 Structure of the work

In section 2 we will give an overview on coordinated multiple views and focus on visual analytics as well as massive, geographical data. Existing concepts and implementations of coordinated multiple views are examined and summarized in an overview.

2 State of the art

Data visualizations are a key part in data-driven decision support systems[14][18]. Few [7] mentions sense-making (also called data analysis) and communication as some of the most important purposes of data visualization. Statistical information is abstract and in data visualization "we must find a way to give form to that which has none."[7]

Visualizations are an obvious choice for managers who demand a quick overview on performance data. In fact Kusinitz [13] explains that the human brain processes visual information 60,000 times faster than text and visual content makes up even 93% of all human communication. Data visualizations are essential here, as managers often do not have the resources to do an in depth analysis with the numbers only. We can expect to see these technologies more in more in business applications. McAfee and Brynjolfsson [17] from the MIT Center of Digital Business showed that organizations driven most by data-based decision making had 4% higher productivity rates and 6% higher profits. However, little research has been done regarding the performance of coordinated multiple views in the field of decision making. There might be a great potential. Back in 1997 Mayer [16] conducted eight studies to compare the effect of using multimedia on university students. The studies showed that when using combined visual and verbal explanations the generation of creative problem solutions increased by an average of more than 50%.

So the application of combined data visualization techniques in decision making seems to be a promising strategy. Nevertheless is is unclear, which visualization techniques are the most suitable to be used in combination. If we know what kind of data we are dealing with, what are the best suited visualization techniques? Let's say we have multidimensional data, is there an order in how people access these multiple dimensions? How do these visualizations perform and what are best practices to be considered for their implementation?

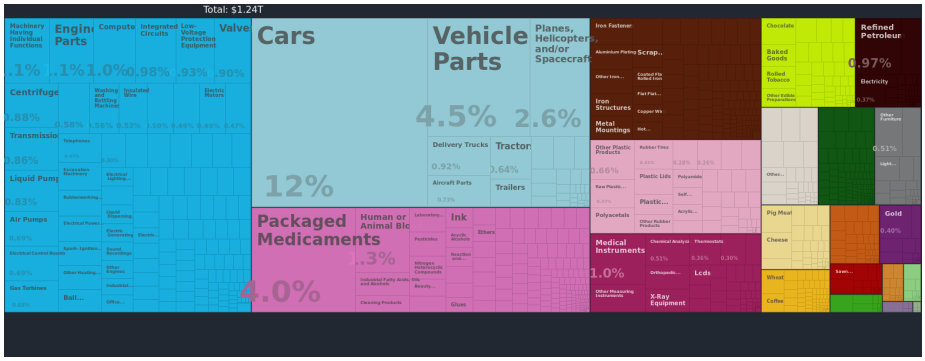
2.1 Data related challenges

Add independent section and explain tasks like data cleansing or data reliability regarding different sources here

2.2 Visualization of hierarchical data

Due to the hierarchical nature of the use cases, we focus on the visualization of hierarchical and geographical data The visualization of hierarchical data has a long tradition. The traditional representation of a tree is a rooted, directed graph with the root node at the top. An everyday use case is a directory tree example of a file system, e.g. in file browsers or command line utilities like `tree` in UNIX. As Shneiderman [21] mentions, this visualization becomes increasingly large when displaying more than one node and soon exceeds the entire screen size. Johnson and Shneiderman [11] proposes the tree map visualization

technique, in which each node is a rectangle whose area is proportional to some attribute, thus making 100% use of the available screen size. As we can see in figure 1 large boxes are labeled with generic tems like “cars” and “medicaments” and include smaller boxes with more specific meanings. We apply the same rules to ordinary maps. The world can be divided into continents, which can be divided into countries, which can be divided into provinces and so on. The difference is that there is no predefined algorithm for layouting, which brings up one of the major disadvantages of tree maps: As the order of placement depends on the respective features of the nodes, small changes in the input data can lead to a large change in the layout of the resulting tree map.



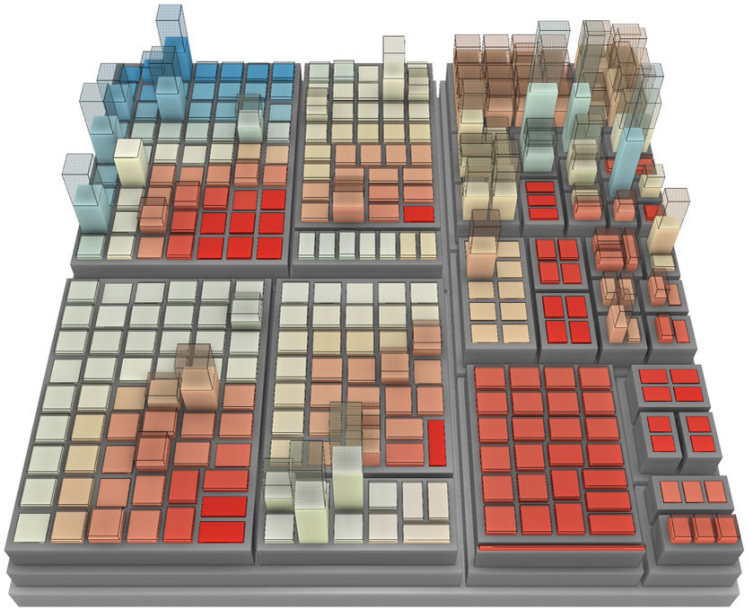
Observatory of Economic Complexity[8]

Figure 1: German exports visualized as a tree map

2.2.1 3D tree maps and 2.5D tree maps

In 2004, Bladh, Carr, and Scholl [2] transfer the concept of tree maps from two dimensional into three dimensional space. The introduce StepTree, which is a three dimensional tree map to display a directory layout. It “differs from Treemap in that it employs three dimensions by stacking each subdirectory on top of its parent directory.”[2] 3D tree maps are superior to 2D tree maps for tasks with a pronounced topological challenge. User perform significantly better in interpreting the hierarchical structure. However, 3D visualizations also introduce some disadvantages as superimposition of objects and a complex view point navigation.

Limberger et al. [15] introduce the concept of a 2.5D tree map which is a constrained 3D tree map. A 2.5D tree map has all items attached to the ground. For the rest of this thesis, we will refer to this type of tree map. We can see an example of a 2.5D tree map in figure 2



© Hasso-Plattner-Institut[5]

Figure 2: Example of a 2.5D tree map

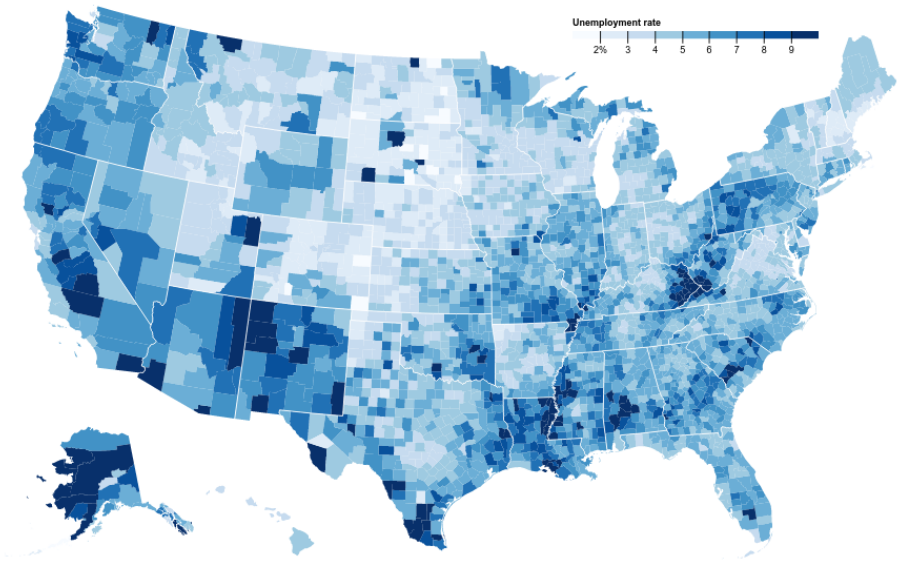
2.3 Choropleth map

A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable being displayed on the map. A popular use case is the display of population density or per-capita income. We can see an example of a choropleth map in figure 3. Choropleth maps are extremely popular and so the audience is likely to understand them. They are very helpful when data is attached to enumeration unites like counties, provinces and countries.

How do tree maps relate to geographical data?

2.4 Coordinated multiple views

According to Roberts [20] coordinated multiple views is just "a specific exploratory visualization technique that enables users to explore their data". Coordinated multiple views are characterized by the fact, that they show multiple views side-by-side. Most multiple coordinated views also provide some kind of brushing technique. "The technique of brushing is the principle approach, where elements are selected (and highlighted) in one display, concurrently the same information in any other linked display is also highlighted."[20] We can see an example in figure 4. It displays an on-time performance of airlines, visual-



GPLv3 Mike Bostock[3]

Figure 3: Unemployment rate in the USA

ized with the "Crossfilter" javascript library. The user can set the borders of an interval with the mouse in each of the views. The visualization takes the most recent 80 flights from the database that match all given filters. All visualizations are then updated in real time. As we can see in the example in figure 4 there seems to be a correlation of a long delay with a later time of the day.

2.4.1 Key interaction techniques for coordinated multiple views

Yi, Kang, and Stasko [26] lists seven categories of interation in information visualization:

1. **Select** mark something as interesting
2. **Explore** show me something else
3. **Reconfigure** show me a different arrangement
4. **Encode** show me a different representation
5. **Abstract/Elaborate** show me more or less detail
6. **Filter** show me something conditionally
7. **Connect** show me related items

Give examples for every category

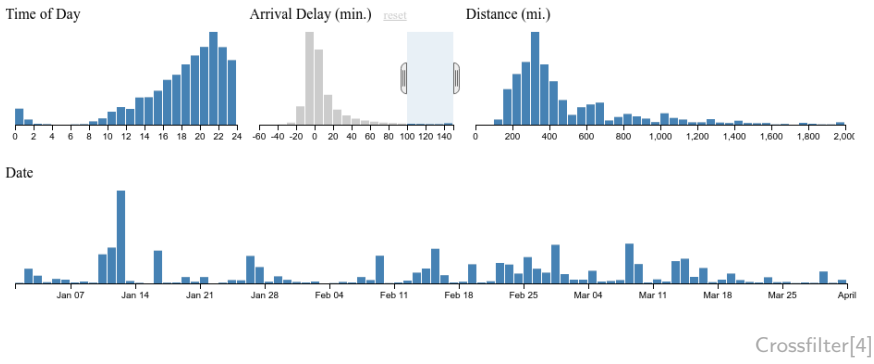


Figure 4: Airline on-time performance: Correlation of time of day with arrival delay. Most recent flight with a delay of more than 100 minutes selected.

3 Analysis

No section without text

3.1 Data sets

For our uses cases, we have two different data sets: One data set consists of a user ranking of public broadcasting in Germany, i.e. entities, mostly TV and radio broadcasts, are liked or disliked by people. This data is public data and can be used by media researchers of broadcasting corporations but also targets media journalists and the general audience. The other set, called `RISO`, consists of statistical data from various German administrations and is used by the authorities for urban planning and policy strategies.

Both data sets share some characteristics. The administrative data connects certain features with certain regions of Germany. As Germany is a federal state, larger regions consist of many other smaller regions.

The second one consists of user data that was collected through a web application called Rundfunk MITBESTIMMEN.

3.1.1 RISO

The `RISO` data base is used in by local authorities to get insights about governmental KPIs to assist local and regional decision making. It is a relational database and a part of the data base schema is shown as an ER diagram in figure 5.

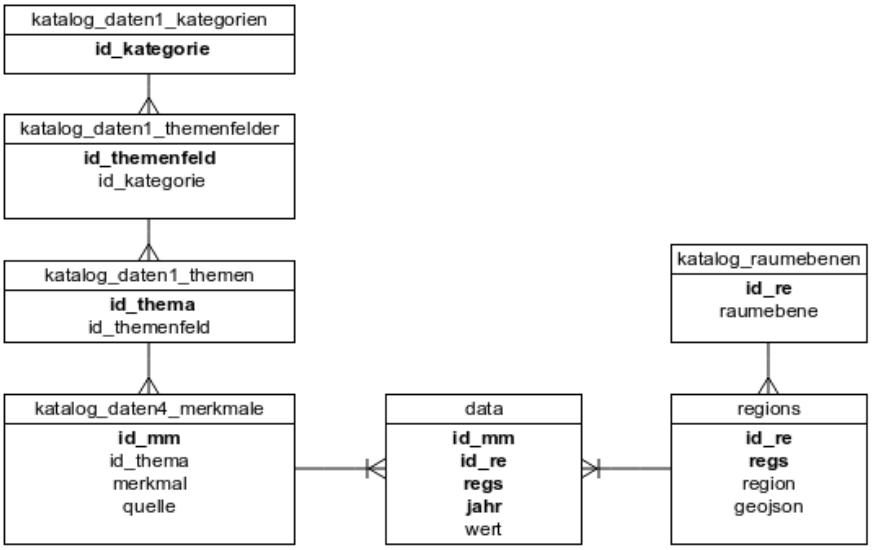


Figure 5: Part of the `RISO` database schema. Primary keys are set in bold.

The largest table is called `data` with approximately 10,466,600 records, which holds all values along with the survey date.

Features This data is connected to a feature table through a foreign key called `id_mm`. In the feature table we can find the description for every referenced feature, e.g. population density, working population in agriculture, education spending. The `RISO` system groups all features in a 4-level hierarchy:

1. `katalog_daten_1_kategorien`
2. `katalog_daten_2_themenfelder`
3. `katalog_daten_3_themen`
4. `katalog_daten_4_merkmale`

The actual features table is the last one in the list. At the lowest level within the hierarchy, this is the largest table with 1234 records.

Regions On the other side, the geographical data is stored in the `regions` table. The geometry data for each region is stored in the `geojson` column and as the name suggests, the data type is a `geojson`. The foreign keys that connect the tables `data` and `regions` are called `id_re` and `regs`. Unlike the feature table, the regions are grouped through the `id_re` that indicates the hierarchy level. So the values of the `id_re` column denominate the level of the hierarchy. E.g. a region with a `id_re` of 1 is a federal state of Germany, a region with id 13 is a constituency. A textual description for the hierarchy level can be found in the `katalog_raumebenen` table in column `raumebene`. Both column `id_re` and `regs` belong to the primary key of the regions table, so there will never be two regions on the same hierarchy level with the same `regs` id.

Characteristics As we can see, the schema of the `RISO` database follows a rather denormalized approach. The schema does not make a lot of assumptions regarding the input data. It allows to add data of arbitrary size, features and completeness as long as there is some kind of numerical data associated with some kind of geographical unit. This approach is suitable for a data base that incorporates data from different sources, as it is the case with the `RISO` data base.

3.1.2 Rundfunk MITBESTIMMEN

Unlike the `RISO` database, the data base of Rundfunk MITBESTIMMEN is used as persistency layer. For that reason the data base schema follows the requirements of a productive web application.

As outline in section 1.1 Rundfunk MITBESTIMMEN is an evaluation platform for public broadcasting in Germany. First, users vote on broadcasts, i.e. they decide if they want to support broadcast or if they do not want to

support. As a next step, user can then make a priorisation by distributing a virtual, monthly budget among the chosen broadcasts.

Figure 6 shows the data base schema of the application. A `user` is connected to `broadcast` through a `selection`. If the `user` supports some a broadcast, the `response` on the given `selection` will be 'positive'. If the `user` does not wish to support a `broadcast`, the `response` will be 'neutral'. The `user` can allocate virtual money to supported broadcasts. The money will be stored in the column `amount` of the `selection`. The sum of all amounts for one user will never exceed the virtual budget of 17.50€.

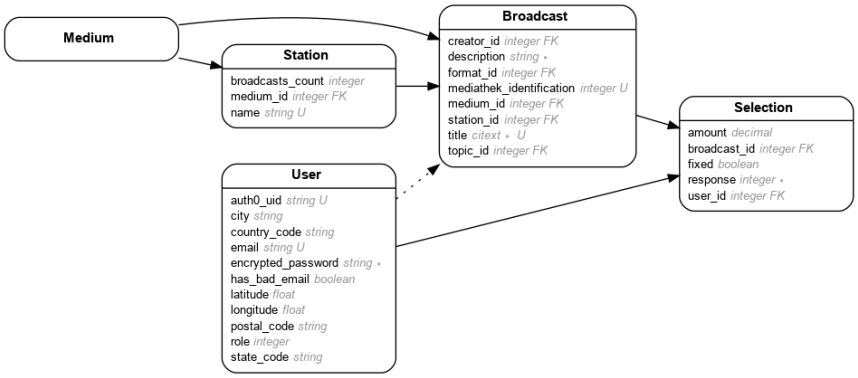


Figure 6: Database schema of the Rundfunk MITBESTIMMEN app

Features We have both numerical as well as nominal features. A numerical feature could be the number of supporters from an area in Germany. A nominal feature could be a list of the most supported broadcasts from an area in Germany. Numerical and nominal features can be combined, so we could request for every region, a distribution of the desired expenditure for radio, TV, online and other broadcasts.

Regions Rundfunk MITBESTIMMEN stores the geometry for each region in `geojson` files. This file holds a `FeatureCollection`. Every `Feature` is a region, the identifier is stored as a property. We merge the geometry data with features for every request. To be precise: We get all the user data, group it by the identifier `state_code` and merge it with the geometry in the `geojson`.

Characteristics The data base schema is a result of the specific requirements of a persistency layer. Changes in the source code may require a migration of the data base schema.

However, we can ask a lot of questions already with common data base queries or standard data analysis tools:

1. How does the actual support of a broadcast compare to the average support of a broadcast?

2. What are the most popular broadcasts in Berlin?

3. What is the desired ratio of genres of supported broadcasts? How important is education compared to sport?

4. How does the support of a broadcast change over time?

5. According to the user ranking, which broadcasts are similar to each other?

3.2 Existing interactions

The possible inteactions in our current VISUAL ANALYTICS PLATFORM can be categorized *select*, *explore*, *reconfigure*, *encode* and *filter*. As seen in figure 7 the user can *select* one item in the view by clicking on it. The user can reveal a tooltip showing the item properties by hovering with the mouse on the item, which is another *selection*. The user can *explore* the map in the usual manner: If the user drags with the mouse on the map, a panning operation is performed with the viewpoint focused on Germany, i.e. the camera moves around like a turntable. The zoom factor can be changed by scrolling on the canvas of the map. *Encode* and *reconfigure* techniques are performed through the menu on the left side: Under the "features" tab, the user can *reconfigure* different data sets and the displayed diagram, e.g. a tree map visualization based on the geometry shape, cubes or voronoi regions. The tab "Dimensions" allows the user to *encode* properties of a data set to visual attributes, e.g. the height, color and texture of an item. The tab "Filter" can be used to reduce the displayed data set along a range of continuous values. Figure 8 shows the range of visible values in the left menu. When the user drags the slider, the items in the map on the right side are updated interactively.

3.3 Interaction to be implemented

We have a focus on coordinated multiple views consisting of a tree map and a geographical map. Let's have some examples how an interaction between a 2.5D tree map and a 2D map might work:

1. User selects a feature set from the drop down in the menu. This will trigger a *Reconfigure* interaction. A data set consisting of all features and their ids, geometries and metadata is transferred. The receiving components are both the 2.5D tree map and the 2D map which will rerender the entire visualization.

2. User hovers with a mouse over a polygon in the 2D map. This will trigger a *Select* interaction. The data is a single feature id that will be transferred to the 2.5D tree map, which will change the color of an box.

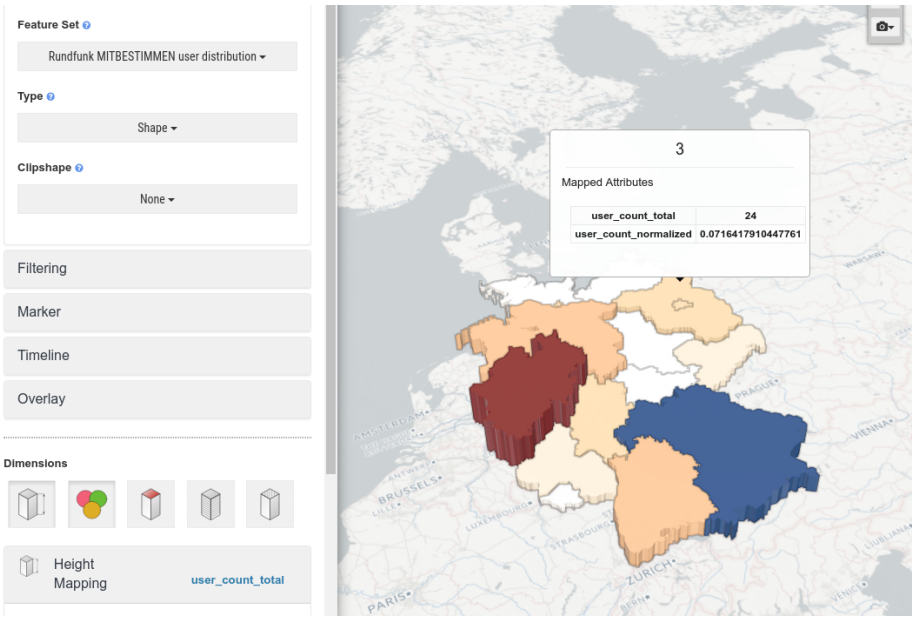


Figure 7: Items can be highlighted with a click, Bavaria is currently highlighted. A mouse over reveals a tooltip showing item properties. The menu on the left side allows to change the data set and the specific base visualization.

3. Rotate or zoom the 2.5D tree map. This will also rotate or zoom the 2D map. The interaction would fall into *Explore* and the shared information is the orientation of the camera and the zoom level.

4. A click in the 2.5D tree map will trigger an *Explore* interaction. The data is a single feature id sent to the 2D map. The map will center the viewport on the center of geometry of the respective feature.

5. The user selects many features at once in the 2D map by dragging a rectangle. The ids of all features within the rectangle are sent to the 2.5D tree map. All features will be highlighted with a different color, which is therefore a *Select* interaction.

6. *Reconfigure* the layouting of the 2.5D tree map by choosing a different hierarchy level. This increased granularity may lead to an increased granularity in the 2D map, e.g. show postal codes instead of federal states. The changed data are additional items, that are nested in the former items.

7. *Encode* the 2.5D tree map by a different attribute mapping like color, height or texture. If the 2D map has no geometry data that defines the shape of a feature, it can also display a larger point marker.

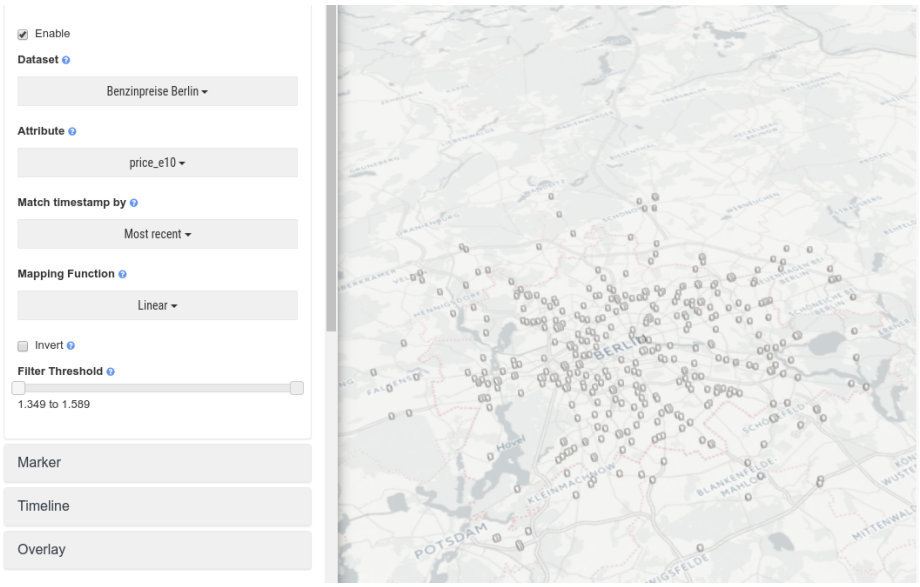


Figure 8: Only gas stations with a price for E10 within 1.349 Euro and 1.589 Euro are displayed in the map

8. Apply a *Filter* and reduce the data set by choosing only items with meta-data beyond a certain treshold. The reduced data leads to a full re-render of all data visualizations. The message contains the updated item list

9. Show a *Connect* by highlighting boxes of the same subtree in the 2.5D tree map. The respective connected items would be highlighted in the 2D map as well. Here the data is a relation between items.

What are the key interactions in our use case?

3.4 Common interface

Having a central controller be responsible for the entire coordination of views can lead to maintainability issues. If a new interaction needs to be implemented, a change in the multiview controller is required. This low flexibility leads to higher development costs and is therefore not desirable.

The question here is: Is there a way to anticipate what kind of data any interaction might affect? What kind of visual changes are possible at all in which types of visualization? What is the expected data structure for which types of visualization? Are there any other shared concepts in all data visualizations?

If there are shared concepts, we might be able to model common interface for all types of visualizations. Multiple views would only need to implement to adapter to this interface. With this interface we would reduce the currently

$N * N$ transformations down to N adapters and reduce the overall development costs.

3.5 Trigger, effect and information semantics

Let's have a closer look what steps a coordinated interaction actually performs. First of all, different visualizations have different capabilities and different degrees of freedom how to interact with and display the data. We choose a *highlight* interaction in our 2.5D tree map as an example. The interaction is started by a low level mouse click event inside the visualization. We call this a *trigger*. The change of the color of a box is perceived by the user. We call this change an *effect*. Both trigger and effect have a certain *semantic* to the user, i.e. the highlighting of the feature id.

We want to coordinte the interaction in multiple views. In order to do so, the semantics need to be exchanged the different views, in our example a *highlight* of a *feature id*.

We will face some issues in coordinating the semantics to other views, because:

1. The effect of the interaction cannot be applied to other views.
2. The relevant information semantics is ambiguous.
3. The low level event that is used as a trigger for the interaction is different.

An example for the first case is to express a highlight interaction with a change of colour. E.g. a bubble chart might encode a certain data attribute to the colour of a bubble. Therefore the colour cannot be changed. Another example for the first case: The data sets of a bar chart could be ordered according to a user interaction. But the ordering cannot be reflected with a change of position in a scatter plot, as the position is tied to a data attribute.

For the second case, we can see an ambiguity of information semantics if we look at line diagrams and a bar charts. A user might click on a line in the line diagram to highlight the *entire series of data*. But in a bar chart a user would click on a bar to select a *feature* of a series, not the entire series of data.

A geographical map might use the mouse drag event to move the viewpoint, which would be an *explore* interaction. But the drag of a mouse could be used in an activity diagram to change metadata, like the start and end time of an activity.

It is possible to describe interactions in any type of data visualization by looking on the *trigger*, the *effect* and the exchanged *information semantics*. Coordinating interactions across multiple views then means to make a reasonable choice of *trigger* and *effect* for every view and then define what data is included in the *information semantics*.

3.5.1 Free visual variables allow for interaction effects

In order to communicate the interaction to the user, the effect of the visualization must be communicated back to the user by a visual change. Most of data visualizations are defined in a way that ties one or more visual variables to the data. These constrained visual attributes are *fixed* while those visual variables that are not tied to the data are therefore *free* to use for a visual feedback of an interaction. Figure 9 shows the seven visual variables introduced by Bertin [1]. These visual variables are used in cartography but can also be applied to data visualization in general.

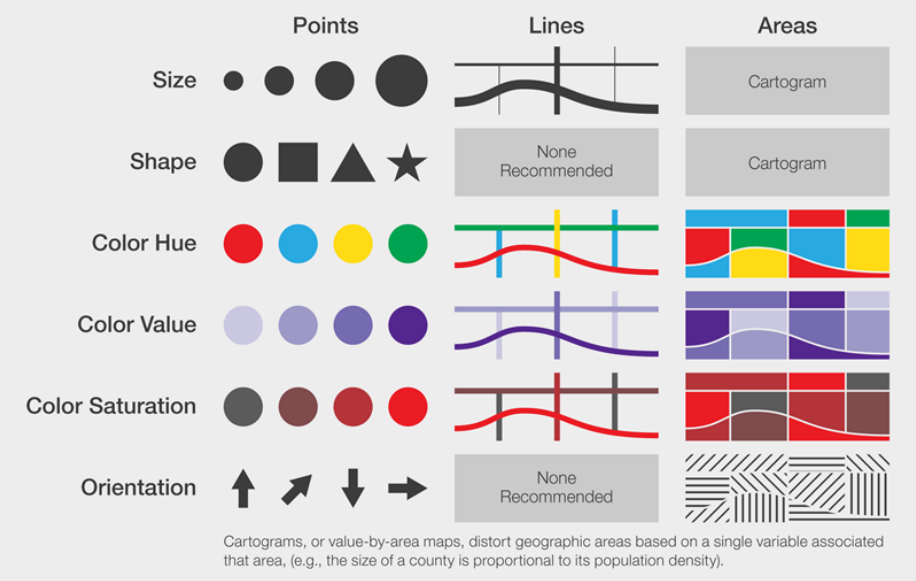
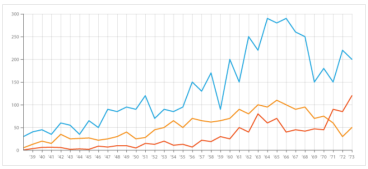


Figure 9: Bertin's[1] original visual variables.

3.5.2 List events as candidates for interaction triggers

Interaction can be triggered by any input of a human-computer-interaction device, like the keyboard or the mouse. Because users are in the habit of expecting certain interactions to be triggered from certain events, we also have constraints here. E.g. the viewpoint in geographical maps is moved by mouse drag events and the zoom level is controlled by the mouse wheel. So we should not use those already occupied events as triggers for our coordinated interactions. Nevertheless, consistency in user interfaces is crucial for great user experience, because users can reuse the learned knowledge. Therefore, we need to list all availabe events as candidates and connect them to our interactions in a preferrably consistent manner.

On the other hand, we might have special controls in visualizations, in order



(a) Line diagram

Figure 10: Line charts are used to display trends

to change the encoding. E.g. a 2.5D tree map shows a menu where the user can define how attributes are mapped to visual variables.

3.5.3 Specify the information semantics

The semantics of a interaction can be manifold. *Select*, *Explore*, and *Filter* interactions affect data items. For that reason, the semantics of the interaction is the type of interaction plus the identifier of the affected data item. A highlighting of a bar in a bar chart would consist of the interaction type *select* and the respective *feature id*. But sometimes, groups of data are selected, like an entire series of data or a subtree in case of hierarchical data.

Reconfigure and *encode* interactions do not affect data items but the data visualization itself. The semantics of these interaction can be arbitrary, e.g. an ordered list of feature ids to define the order or a data object that models the mapping of data attributes to visual variables.

Abstract/elaborate and *connect* interactions are special interactions which expect a certain type of data. Increasing the level of detail expects a tree of data with the level of detail determining the maximum depth of the visualized tree. Showing connections between features expects a graph of data, with edges between features and properties on these edges.

3.6 Interaction examples in other visualizations

The data visualization cataloge by Severino Ribecca list many of the most used data visualizations[19]. Let's pick some of those and derive a number of possible interactions.

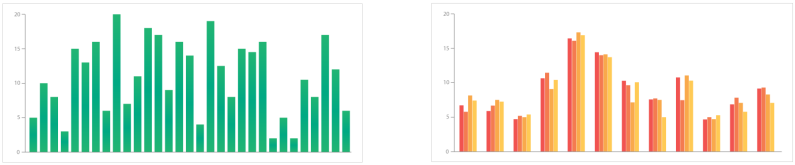
3.6.1 Line diagrams

Line diagrams are multiple sets of data, displayed along the x-axis. They are used to display quantitative value over a continuous interval or time span. It is possible to highlight an entire series of data or just a feature within that series.

Data structure	Tabular data, many data sets as series.
Fixed visual attributes	Position, orientation, texture.
Free visual attributes	Color, shape, size.

Figure 11: Interactions for line charts

Select	Highlight a data point (id of data point)
Select	Highlight a data series (id of data series)
Encode	Change colours of data series (data series → colour)
Filter	Restrict interval on x-axis (filter function of data attribute)
Filter	Hide a data series (id of data series)



(a) Bar chart (b) Multiset bar chart

Figure 12: A multiset bar charts is a variation of a bar chart

3.6.2 Bar charts and multiset bar charts

Bar charts and multiset bar charts show one or many attributes per feature along an axis. They have in common that they encode the data attribute into the height of the eponymous bars. Colors might be used to better distinuish between the different data attributes. Features can be arbitrarily ordered along the axis, although multiset bar charts group features along a series of categories.

Therefore, bar charts need to be initialized with the set of features and their values as well as the groups of features for the multiset bar chart. The supplied colours would colour each feature in a group in turn. The set of possible ineractions include the highlighting of features, the reordering of features and a change in encoding of colour values.

Data structure	Tabular data, many data sets as series
Fixed visual attributes	Size, orientation.
Free visual attributes	Position, colour, shape, texture.

Histograms visualise the distribution of data over a continuous interval or certain time period. A special type is the population pyramid, which is a pair

Figure 13: Interactions for bar charts

Select	Highlight a bar (id of data point)
Encode	Change colours of data series (colours → data series)
Reconfigure	Sort by attribute (data attribute)
Reconfigure	Drag bars to reorder data series (ordered list of ids of data points)
Filter	Hide a data series (id of data series)

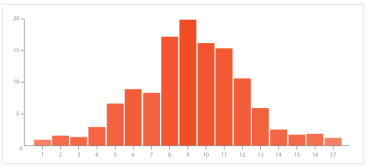
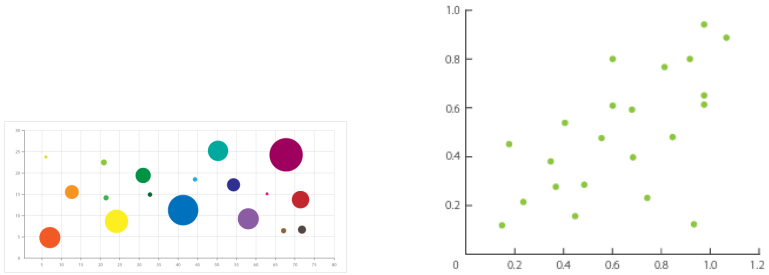


Figure 14: A histogram is a bar chart over a continuous interval



(a) Bubble chart (b) Scatter plot

Figure 15: Bubble charts and scatter plots are similar regarding interactions

of back-to-back histograms, one for each sex. The difference of histograms to bar charts lies in the type of data itself, not the representation. Therefore these charts need to be initialized with the same data and the same interactions can be applied.

Data structure	Tabular data, many data sets as series
Fixed visual attributes	Size, orientation, position.
Free visual attributes	Color, shape, texture.

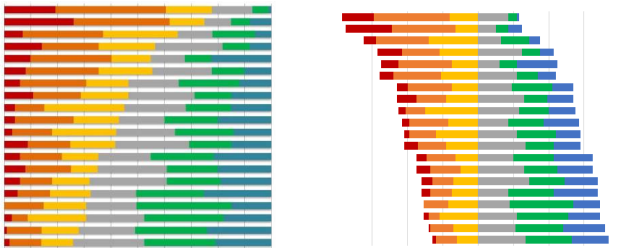
3.6.3 Bubble charts and scatter plots

Bubble charts are popular choices to display a distribution of features. The chart is initialized with coordinates for each feature, a colour and a size in case of a bubble charts. Possible interactions include the highlighting of features, a different colour encoding, a reconfiguration to map another attribute to size. Bubble charts may show only a window of the available data and allow to zoom in, zoom out or move the window along the axes.

Data structure	Tabular data, single data set with x- and y-coordinates
Fixed visual attributes	Size, position.
Free visual attributes	Color, shape, texture, orientation.

Figure 16: Interactions for bubble charts

Select	Highlight a bubble (id of data point)
Explore	Zoom in, zoom out (width and height of window)
Explore	Move viewport position (x- and y-coordinates of viewport)
Encode	Change mapping of colour to category (data series → colour)
Encode	Change colour function (function value → colour)
Encode	Change data attribute to colour (data attribute)
Encode	Change data attribute to size
Reconfigure	Sort by attribute (data attribute)
Reconfigure	Drag bars to reorder data series (ordered list of ids of data points)
Filter	Hide a data series (id of data series)



(a) Stacked bar chart

(b) Stacked bar chart with baseline

Figure 17: Stacked bar charts can be ordered along a baseline or stretch to 100% width to show the percentage-of-the-whole of each group

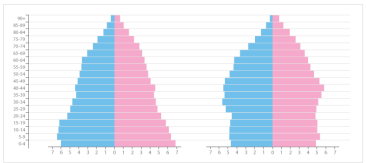


Figure 18: A population pyramid can be modeled as a stacked bar chart

Figure 19: Interactions for stacked bar charts

Select	Highlight a bar (id of data point)
Encode	Change mapping of category to colour (data series → colour)
Reconfigure	Sort by attribute (data attribute)
Reconfigure	Reorder Y axis (ordered list of ids of data points)
Reconfigure	Sort stacking order by attribute (data attribute)
Reconfigure	Specify the stacking order data series (ordered list of ids of data series)
Reconfigure	Specify a negative data series (list of ids of data series)
Filter	Hide a data series (id of data series)

(a) Tree map (b) Sunburst diagram

Figure 20: Tree maps and sunburst diagrams are ideal to show hierarchies

3.6.4 Stacked bar charts

Unlike a multi-set bar graph which displays their bars side-by-side, stacked bar graphs segment their bars of multiple datasets on top of each other. A baseline, as shown in figure 17 might be modeled as two back-to-back multi-set bar graphs. A reordering would e.g. move one data set from the left side to the right side. Possible interactions include the highlighting of a feature, a change of color mapping, reordering of the baseline.

Data structure	Tabular data, multiple date sets as series
Fixed visual attributes	Size, shape, orientation.
Free visual attributes	Color, position, texture.

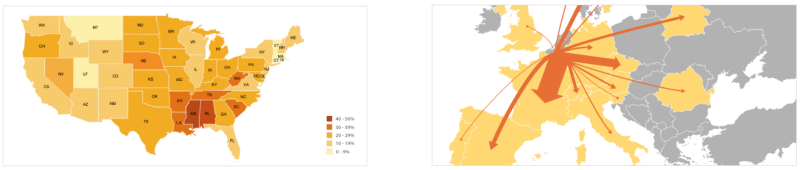
3.6.5 Hierarchical data

Treemaps are great to show hierarchical data without ever exceeding the availabe screen. Each feature is a assigned a rectangle according to a layouting algorithm. Unlike a tree map a hiearchical ring diagram or sunburst diagram shows each level of the tree as a series of rings.

Therefore, both tree map and ring diagram need the feature set as a tree, with each node having a data attribute for layouting. Every node may be assigned a color. As we are describing hierarchies, the maximal depth of tree may be increased or decreased. Again, interactions could include a highlighting of features and a change of color encoding. Both visualizations may show only a subtree. E.g. a click on a box in the treemap opens another treemap focused

Figure 21: Interactions for hierarchical visualizations

Select	Highlight a feature (id of data point)
Explore	Use another node as root of the visible tree (id of data point)
Encode	Change mapping of category to colour (data series → colour)
Reconfigure	Change data attribute used for layouting (data attribute)
Reconfigure	Sort by attribute (data attribute)
Reconfigure	Specify order (ordered list of ids of data points)
Abstract/Elaborate	Specify maximum depth of visible tree (number of levels)



(a) Choropleth map (b) Flow map

Figure 22: Choropleth maps focus on a density while flow maps show a migration of data

on the subtree. Similarly a click on a slice of the ring would surround the most external ring with the children of the feature.

Data structure	Tree, each feature has a value for layouting.
Fixed visual attributes	Position, Size, shape, orientation.
Free visual attributes	Color, texture.

3.6.6 Geographical data

Choropleth maps and flow maps are specialized diagrams focused on geographical data. Size, position and shape of a feature is defined by the geometry data of a feature. In choropleth maps the color of each feature is based on a data attribute. Flow maps may display connections between features, a data value defining the size of each arrow.

Data structure	Graph data with edges, each feature has geometry data.
Fixed visual attributes	Position, Size, shape, orientation.
Free visual attributes	Color, texture.

3.6.7 Activity diagrams

In activity diagrams, each feature is represented as a rectangle, with the duration of the activity mapped to size and position. Calendars and gantt charts could not only read the data from the data source, but also add new features to the data set or update metadata of a feature, e.g. the progress of the activity.

Figure 23: Interactions for geographical visualizations

Select	Highlight a feature (id of data point)
Explore	Move viewport (latitude and longitude of viewport)
Explore	Zoom in, zoom out (zoom factor)
Encode	Change shape of marker (data id → shape)
Encode	Change mapping of category to colour (data series → colour)
Encode	Change colour function (value → colour)
Encode	Change data attribute used for colour (data attribute)
Connect	Show relations of a feature (id of data point)
Abstract/Elaborate	Change granularity of displayed regions (number of levels)

(a) Calendar (b) Gantt chart

Figure 24: Similar to a calendar, a gantt chart shows activities and the progress along a time line

Data structure	Temporal data, each feature has a time interval.
Fixed visual attributes	Position, Size, orientation.
Free visual attributes	Color, shape, texture.

Figure 25: Interactions for temporal visualizations

Select	Highlight a feature (id of data point)
Explore	Show a different period of dates (start and end datetime)
Explore	Show a different time interval (start and end hour)
Encode	Change color of categories or activities (data series → colour)
Encode	Change data attribute used for colour (data attribute)
Filter	Remove a calendar or a category (id of data series)

4 Concept

No section without text

4.1 Abstract data model

To account for the various data structures, we use an abstract data model that is powerful enough to include tabular, hierarchical and relational data. You can see a class diagram in figure 26.

The *entity* class is used to model the smallest distinguishable unit. All entities can be identified and retrieved via the *id*. An entity is defined to be any object that can have data attached as attributes. It can have arbitrary many attributes and each value can be accessed by the name of the attribute. So if you want to get the *latitude* value of an entity, you can retrieve the value with a call to the named attribute *latitude*.

Entities can also be *series* of other entities. A series contains an ordered list of contained entities. As series can also contain other series, so we can model a hierarchy relation.

There is always one particular series without a parent. This is the root node of the hierarchy. If we just want to display tabular data, we just have one or two levels of hierarchy. E.g. one level of hierarchy for a histogram and two levels of hierarchy for a stacked bar chart.

Other relations than hierarchical relations can be modeled as a *relation* entity. It represents a directed edge in a graph and must have incoming and outgoing entity. Since every *relation* is an *entity* as well, we can add *attributes* to the relation. These attributes may describe e.g. the weight of an edge in a flow map.

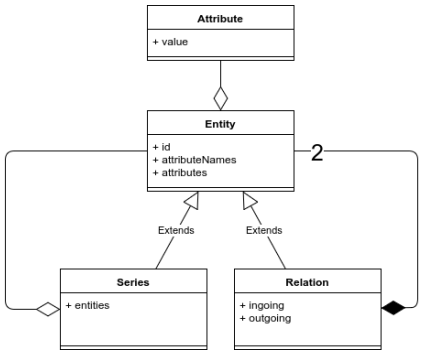


Figure 26: A data structure for tabular, hierarchichal and relational data

4.2 Interaction semantics expressed as functions

It turns out that we can describe the semantics of an interaction as a mathematical function. These functions operates on the ids of *entities*, *series*, and *relations* or their respective *attributes*. The change of the function describes the interaction but ignores implementation details of specific visualizations. To coordinate interactions across views, one view sends the new function as a message to its companioning views.

The function declarations are derived from specific interactions of the examples in section 3.6. We describe them as follows:

1. $Highlight : \varnothing \rightarrow \mathbb{N}$

   A highlighting is a function with no input parameters that is returning a list of highlighted entities

2. $Filter : \varnothing \rightarrow \mathbb{N}$

   The $Filter$ function takes no arguments and returns a subset of entities that should be displayed.

3. $FilterByValue : \mathbb{N}^* \rightarrow \{\bot, \top\}$

   The $FilterByValue$ function returns whether an entity belongs to the displayed set based on a list of attribute values.

4. $FilterAttributes : \varnothing \rightarrow \sum^*$

   The $FilterByAttributes$ returns an arbitrary long, ordered list of names of attributes that are taken as input for the $FilterByValue$ function.

5. $Focus : \varnothing \rightarrow \mathbb{N}$

   The $Focus$ function returns a single id of the entity that should be in the center of the viewport. A tree map or a sunburst map can use the focused entity as the root node of the visible subtree.

6. $Screen : \varnothing \rightarrow \mathbb{N}*$

   This function returns a list of entities that must be in between the current boundaries of the diagram window. The list can be used to implicitly derive the zoom level in different visualizations like bubble chart or geographical map. It can also be used to implicitly derive the visible hierarchy levels in a tree map.

7. $Colour : \mathbb{N} \rightarrow \mathbb{C}$

   A colour function assigns an entity or a data series to a color c with $c \in \mathbb{C}$ the set C of all colours. A colour typically consists of three values R, G and B.

8. $ColourByValue : \mathbb{R} \rightarrow \mathbb{C}$

   A colour function assigns a continuous value to a color c with $c \in \mathbb{C}$ the set C of all colours This function is e.g. used by a choropleth map.

9. $ColourAttributes : \varnothing \rightarrow \sum^*$

   The names of the attributes used as input for the $ColourByValue$ function.

10. $Order : \mathbb{N} \rightarrow \mathbb{N}$

    An order function assigning each entity to an index. This index will be used in bar charts to order bars along the x-axis or stacked bar charts, to order the stacking of data series.

11. $Sort : \left(\mathbb{R} \times \mathbb{R}\right)^* \rightarrow \mathbb{R}$

    This sort function will return a continuous value based on an arbitrary long list of pairs of attribute values. It's an implicit order function based on one or many data attributes and returns a negative value if the first element is orderd before the second element.

12. $SortAttributes : \varnothing \rightarrow \sum^*$

    The ordered list of names of attributes which will be used as input for the $Sort$ function.

13. $X : \varnothing \rightarrow \sum$

    The X function denotes the name of the attribute used for the x-axis of the diagram. In bar charts, the categories are displayed along the x-axis. In a population pyramid the x-axis is the age of people.

14. $Y : \varnothing \rightarrow \sum$

    The Y function denotes the name of the attribute used for the y-axis of the diagram. Bar charts map a data attribute to the length of a bar along the y-axis.

15. $Size : \varnothing \rightarrow \sum$

    This function denominates the name of the attribute used to encode the $Size$ of the feature. Bubble charts can encode the data attribute to the area of the bubble.

16. $LayoutAttribute : \varnothing \rightarrow \sum$ This specialized function returns the name of the attribute that should be used for layouting in hierarchical visualizations.

17. $Shape : \mathbb{N} \rightarrow \mathbb{S}$ For visualizations, that display a shape per data point, $Shape$ will return a shape $s \in S$ with S the set of all discrete shapes.

Let's have some examples how these functions can be applied on coordinated interactions:

A user clicks on a bar in a bar chart and this feature then changes its background colour. To coordinate the highlighting, the bar chart visualization replaces the *Highlight* function. The function now returns the id of the entity of the feature with the new background colour.

When a region in a geographical map is clicked, the new *Focus* function will return the id of the clicked entity. A coordinated tree map next to the geographical map now shows a subtree with the focused entity as root node.

If many attributes are chosen from a dropdown menu and some thresholds are specified with a slider, both the *FilterAttributes* and the *FilterByValues* functions are changed. The *FilterAttributes* will return the ordered list of chosen attributes of the dropdown menu. The updated *FilterByValues* function will now expect a new argument list, based on $FilterAttributes$. It then yields either true or false based on the values of these attributes.

4.3 Publish and subscribe to interactions

In section 4.2 we discussed how we can describe the semantics of the interaction. We still need to describe how we can model the coordination of the interaction, i.e. the actual message flow of interactions. Obviously, the semantics is somehow the payload of a message. But what is the framework to exchange the these messages? How is one action in one view coupled with a visual effect in another view? Figure 27 shows a model of the coordination. We introduce two new

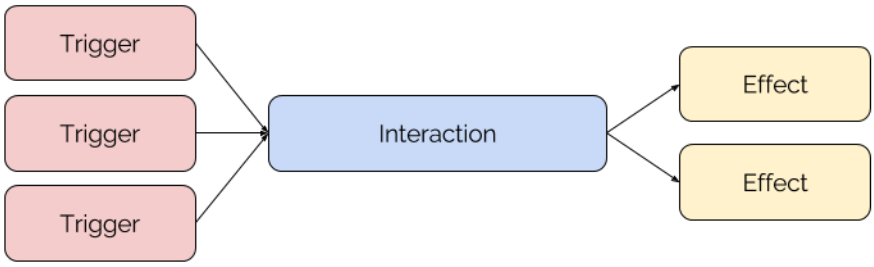


Figure 27: Schema of message flow in coordinated multiple views

terms, *trigger* and *effect*.

A trigger is e.g. a mouse interaction, like hover or click. This low-level action is an implementation detail and therefore tied to the specific visualization.

In order to be perceivable by the user, the interaction must have some visual effect. A feature of the visualization is changed, e.g. a colour of a bar in a bar chart. The choice of the visual effect to express the interaction is an implementation detail as well.

Sometimes a visualization may not be able to interpret an interaction. E.g. a bar chart can re-arrange the bars along the x-axis in case of a *Reconfigure* interaction. But a scatter plot constrains x- and y-coordinates of an entity on a certain data attribute. Therefore, not only the *trigger* and *effect* is implementation specific, but also the handling of the interaction itself.

Every visualization decides on its own, how to react to a certain interaction. That leads us to distinguishable, named interactions. Every view can subscribe

to certain interactions and receive messages in form of changed interaction semantics. In order trigger an interaction, the visualization simply publishes to the named interaction.

This pattern is known as the *Publish-subscribe pattern* and widely used in message queues. The term *interaction* in our case is equivalent to the term *channel* or *topic* commonly used in message queues.

5 Implementation

This section describes the implementation details for the described concepts. We start with a list of to be implemented features. Next, we describe the architecture of the software and why we chose this architecture. What requirements and considerations lead to this particular architecture.

5.1 Implemented interactions

In the course of this thesis we want to implement the following interactions:

- *Select*: The user clicks on a building or region in a geographical map and all affected properties in the 2.5D tree map will be highlighted.
- *Explore*: The user clicks on a block in the 2.5D tree map and the viewpoint in the geographical map will be centered on relevant area.
- *Abstract/Elaborate*: The user selects a different granularity in the geographical map (e.g. postal code regions instead of federal states) and the change is reflected in the 2.5D tree map.
- *Filter*: The user double-clicks on a region in a geographical map and the 2.5D tree map will be based on data of only that region.

5.2 Observer pattern

Explain observer pattern

Figure 28 shows how coordinated multiple views can be automatically updated even if the environment lacks a native update-mechanism. We use the observer pattern: The root of our views is an observable implementation of the common interface described in section4. If one of the views updates the view state, it will send the necessary information to the observable. The observable will then broadcast the change to all it's observers.

Publisher subscriber In our particular case we apply a special form of the observer pattern, the so called "Publish-subscribe" pattern[6]. Publish-subsribe is a messaging pattern which is widely used in message queues. In this scenario, senders of messages simply categorize their messages which will be consumed by subscribers of the category. The scenario has very low coupling, publishers do not even need to know the existence of subscribers.

How can multiview subscribe to the multiview controller

5.3 Component pattern

State-of-the-art javascript frameworks like ReactJS and EmberJS follow the component pattern for the architecture of a single page web application. The

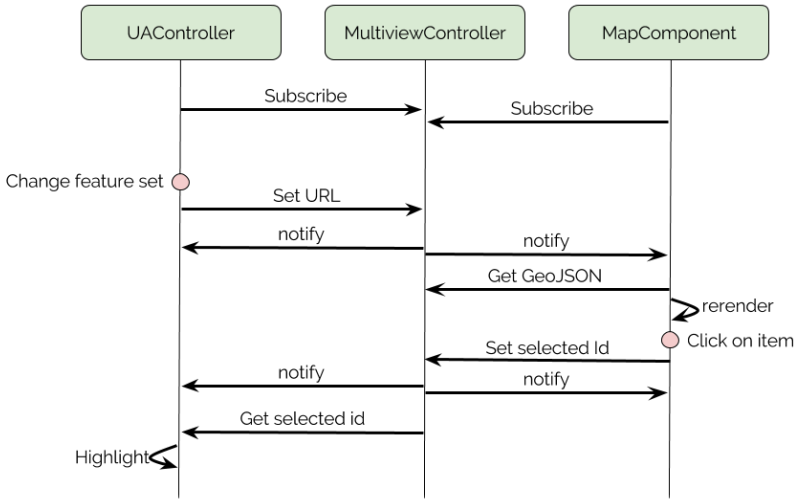


Figure 28: The sequence diagram shows the notification of different components. The user first chooses a feature set and hovers over a polygon in the geographical map.

component pattern imposes a hierarchical structure on a website. Each component is responsible for a task and may contain other components. The components are joined at the root node of the page.

This pattern is very applicable to coordinated multiple views. The different views of coordinated multiple views share state, i.e. the feature, that is currently highlighted or the applied filter on the data. So the views are components and their closest common ancestor is the coordinated multiple view itself, controlling state and passing user interaction down to it's children.

Actions up - Data down Version 2.0 of Ember introduced a common phrase how to use this pattern effectively: "Data down, actions up"[22] In the domain of coordinated multiple views actions would mean user interactions, e.g. a click on a feature. The action will notify the controlling coordinated multiple view component. Actions may change data, and the changes will be passed to to all dependen views. These views are then rerendered.

Examples for the kind of data that might trigger a rerendering of a view:

- The selected feature or a list of selected features
- A list of thresholds for certain features as a filter

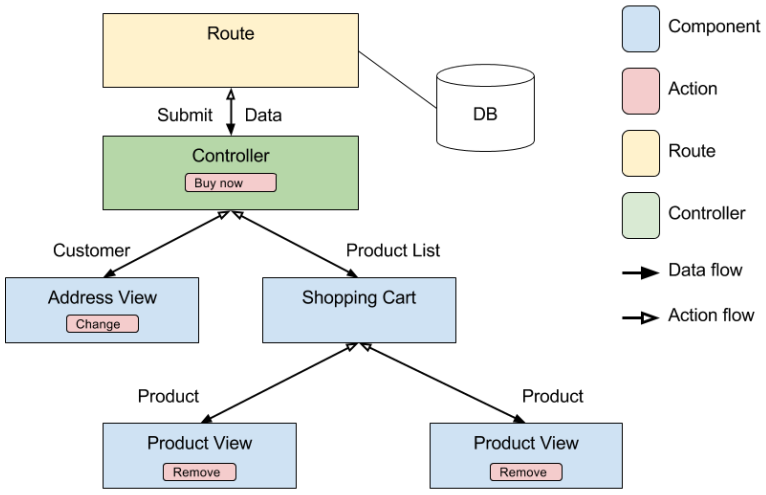


Figure 29: Implementation of the component pattern in EmberJS. The example shows a page of a webshop. The customer is about to order the items in the shopping cart.

5.4 Web components W3C specification

Web components is a recent standard of the W3C[23] to bring component-based software engineering to the world wide web. They look perfectly suited to be used in coordinated multiple views. However the attributes of web components are string based. If arbitrary javascript objects need to be passed into the web component it is suggested to use one of the common javascript frameworks that allow for data binding.

5.5 Frontend framework comparison

We evaluated three javascript frontend frameworks for our application: *GlimmerJS*, *Google Polymer* and *ReactJS*. Figure 30 shows the pros and cons of each framework for our use case.

GlimmerJS is the rendering enginge of EmberJS[25]. In 2017 it was released as a standalone framework. Applications written in GlimmerJS can be exported as web components. These web components can be included in any website, which makes GlimmerJS a reasonable choice to build high-quality widgets for user interfaces. GlimmerJS also uses handlebars[12], a user-friendly templating

language. The downside of GlimmerJS is the current lack of documentation and immaturity due to the recent first release this year.

Another popular framework to build web components is Google's Polymer library[10]. With 18,469 stars on Github it is the most popular framework for web components at the time of writing. Polymer has a large community and comprehensive documentation and therefore more suitable than GlimmerJS for our task.

Unfortunately, the web component standard does not specify how arbitrary Javascript objects can be passed to web components. This raises some problems in legacy apps: Usually, legacy apps are written in plain javascript without the use of a component-based frontend framework. Any part of the code may call any other part of the code, leading to the dreaded "spaghetti code". Refactoring the existing app requires the framework to have a reliable way of communication with the legacy parts. E.g. parts of the legacy code call the backend in order to load data. This data needs to be passed to the compponents of the user interface. Web components do not have a designated interface Because of that, libraries like Polymer come back on proprietary solutions. But those proprietary solutions defeat the main advantageof developing against a standard, as it is unclear how components may interact with each other.

Lifting the constraint to implement against web components, we finally decided to go for ReactJS[9]. ReactJS does not have the ability to export web components. It has, in return, an ascertained way of integrating the framework into a legacy app built with jQuery. Along with its major advantage of easy integration, it has a striving community, heaps of documentation and tutorials and is well tested.

All of these reasons make us choose ReactJS for the task of coordinating multiple views in our existing VISUAL ANALYTICS PLATFORM.

Pros/cons of GlimmerJS, Polymer, React

5.6 Observer pattern and component pattern in coordinated multiple views

Figure 31 shows the final result. We try to put as much code as possible under the root node of ReactJS. By that we eliminate the amount of custom updating implementation. The root node of the DOM-tree of our react application is connected with the existing app through the common interface. Both urban analytics controller and the multiview map component will observe changes to the common interface. Also sub-components may communicate with the common interface.

5.7 GeoJSON

An example of aggregated user data merged with geometry data can be seen in listing 1.

```
1 {
```

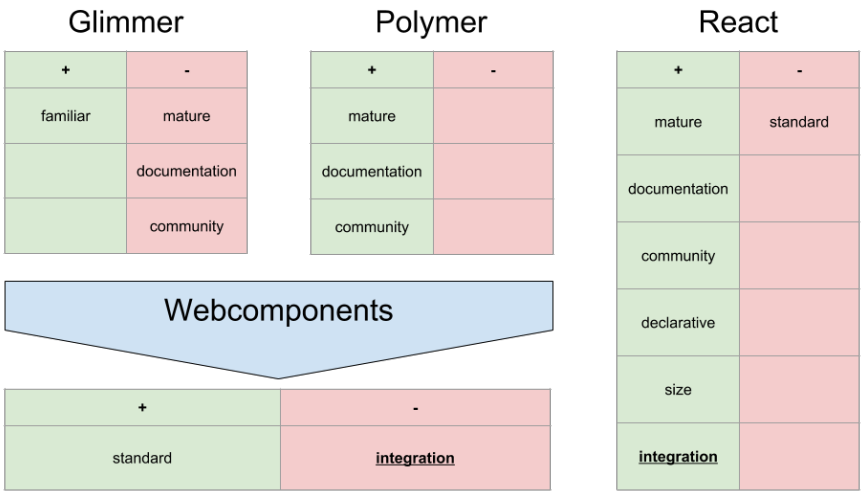


Figure 30: Comparison of frontend frameworks

```
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": []
      },
      "properties": {
        "NAME_1": "Baden-Württemberg",
        "state_code": "BW",
        "user_count_total": "34",
        "user_count_normalized": "0.10149253731343283"
      },
      "id": 0
    },
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": []
      },
      "properties": {
        "NAME_1": "Bayern",
        "state_code": "BY",
        "user_count_total": "36",
```

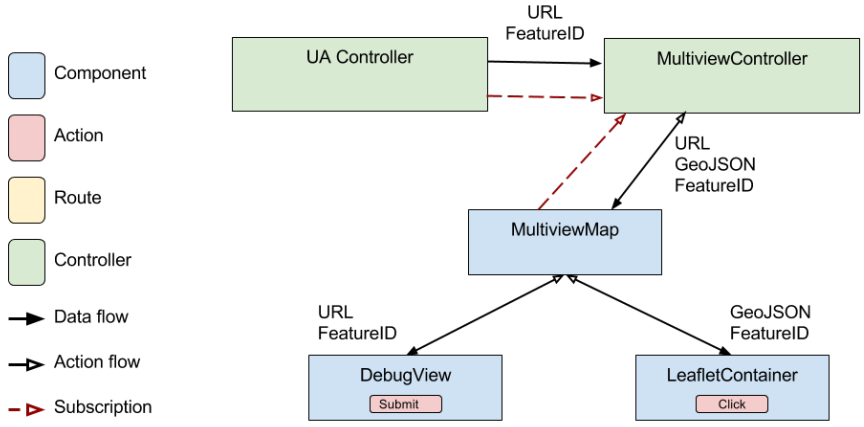


Figure 31: Both observer pattern and component pattern applied in the field of coordinated multiple views

```
        "user_count_normalized": "0.10746268656716418"
      },
      "id": 1
    }
  ]
}
```

Listing 1: Geojson example

We can use this data as input for our common VISUAL ANALYTICS PLATFORM, e.g. figure 32 shows the user distribution of RUNDFUNK MITBESTIMMEN.

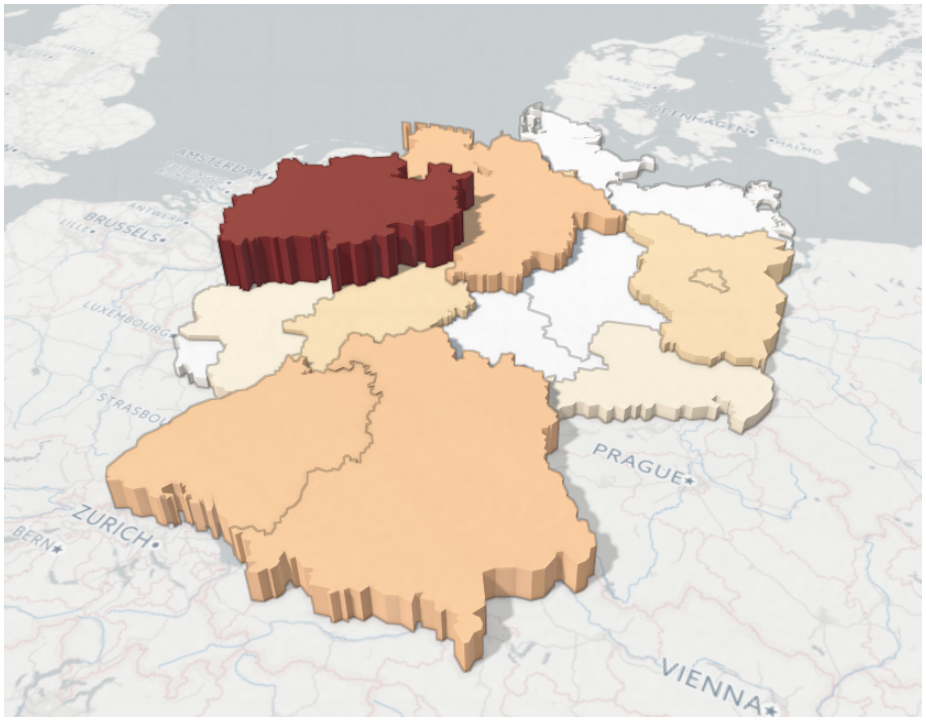


Figure 32: User distribution of RUNDFUNK MITBESTIMMEN across German federal states

References

[1] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. Jan. 2010. ISBN: 978-1-58948-261-6.

[2] Thomas Bladh, David A. Carr, and Jeremiah Scholl. "Extending Tree-Maps to Three Dimensions: A Comparative Study". In: *Computer Human Interaction: 6th Asia Pacific Conference, APCHI 2004, Rotorua, New Zealand, June 29-July 2, 2004. Proceedings*. Ed. by Masood Masoodian, Steve Jones, and Bill Rogers. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 50–59. ISBN: 978-3-540-27795-8. DOI: 10.1007/978-3-540-27795-8_6. URL: http://dx.doi.org/10.1007/978-3-540-27795-8_6.

[3] Mike Bostock. *Choropleth*. July 9, 2017. URL: https://bl.ocks.org/mbostock/4060606 (visited on 07/10/2017).

[4] Mike Bostock. *Crossfilter*. July 2017. URL: http://square.github.io/crossfilter/ (visited on 07/10/2017).

[5] Jürgen Döllner. *Visualization Techniques for Big Data*. July 2017. URL: https://hpi.de/doellner/masterarbeiten/visual-software-analytics.html (visited on 07/11/2017).

[6] Patrick Th. Eugster et al. "The Many Faces of Publish/Subscribe". In: *ACM Comput. Surv.* 35.2 (June 2003), pp. 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: http://doi.acm.org/10.1145/857076.857078.

[7] Stephen Few. "Data visualization for human perception". In: *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* (2013).

[8] Macro connection group. *The Observatory of Economic Complexity*. July 2017. URL: http://atlas.media.mit.edu/en/profile/country/deu/ (visited on 07/03/2017).

[9] Facebook Inc. *React, A javascript library for building user interfaces*. Sept. 7, 2017. URL: https://facebook.github.io/react/ (visited on 09/23/2017).

[10] Google Inc. *Polymer Project*. Sept. 22, 2017. URL: https://www.polymer-project.org/ (visited on 09/23/2017).

[11] Brian Johnson and Ben Shneiderman. "Tree-Maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures". In: *Proceedings of the 2Nd Conference on Visualization '91*. VIS '91. San Diego, California: IEEE Computer Society Press, 1991, pp. 284–291. ISBN: 0-8186-2245-8. URL: http://dl.acm.org/citation.cfm?id=949607.949654.

[12] Yehuda Katz. *handlebars*. June 22, 2017. URL: http://handlebarsjs.com/ (visited on 09/23/2017).

[13] Sam Kusinitz. *12 Reasons to Integrate Visual Content Into Your Marketing Campaigns*. July 18, 2014. URL: https://blog.hubspot.com/marketing/visual-content-marketing-infographic (visited on 07/08/2017).

[14] Nada Lavrač et al. “Data mining and visualization for decision support and modeling of public health-care resources”. In: *Journal of Biomedical Informatics* 40.4 (2007). Public Health Informatics, pp. 438–447. ISSN: 1532-0464. DOI: http://dx.doi.org/10.1016/j.jbi.2006.10.003. URL: http://www.sciencedirect.com/science/article/pii/S153204640600116X.

[15] Daniel Limberger et al. “Dynamic 2.5D Treemaps Using Declarative 3D on the Web”. In: *Proceedings of the 21st International Conference on Web3D Technology*. Web3D ’16. Anaheim, California: ACM, 2016, pp. 33–36. ISBN: 978-1-4503-4428-9. DOI: 10.1145/2945292.2945313. URL: http://doi.acm.org/10.1145/2945292.2945313.

[16] Richard E. Mayer. “Multimedia learning: Are we asking the right questions?” In: *Educational Psychologist* 32.1 (1997), pp. 1–19. DOI: 10.1207/s15326985ep3201_1. eprint: http://dx.doi.org/10.1207/s15326985ep3201_1. URL: http://dx.doi.org/10.1207/s15326985ep3201_1.

[17] Andrew McAfee and Erik Brynjolfsson. *Big Data: The Management Revolution*. Oct. 2012. URL: https://hbr.org/2012/10/big-data-the-management-revolution (visited on 07/08/2017).

[18] Thiago Poleto, Victor Diogho Heuer de Carvalho, and Ana Paula Cabral Seixas Costa. “The Roles of Big Data in the Decision-Support Process: An Empirical Investigation”. In: *Decision Support Systems V – Big Data Analytics for Decision Making: First International Conference, ICDSST 2015, Belgrade, Serbia, May 27-29, 2015, Proceedings*. Ed. by Boris Delibašić et al. Cham: Springer International Publishing, 2015, pp. 10–21. ISBN: 978-3-319-18533-0. DOI: 10.1007/978-3-319-18533-0_2. URL: http://dx.doi.org/10.1007/978-3-319-18533-0_2.

[19] Severino Ribecca. *The Data Visualisation Catalogue*. Sept. 11, 2017. URL: http://www.datavizcatalogue.com/index.html (visited on 09/23/2017).

[20] J. C. Roberts. “State of the Art: Coordinated Multiple Views in Exploratory Visualization”. In: *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007)*. July 2007, pp. 61–71. DOI: 10.1109/CMV.2007.20.

[21] Ben Shneiderman. “Tree Visualization with Tree-maps: 2-d Space-filling Approach”. In: *ACM Trans. Graph.* 11.1 (Jan. 1992), pp. 92–99. ISSN: 0730-0301. DOI: 10.1145/102377.115768. URL: http://doi.acm.org/10.1145/102377.115768.

[22] Frank Treacy. *Getting Started with Ember and Data Down Actions Up*. 2017. URL: https://emberigniter.com/getting-started-ember-cli-data-down-actions-up-tutorial/ (visited on 07/23/2017).

[23] World Wide Web Consortium (W3C). *Web components current status*. Feb. 13, 2017. URL: https://www.w3.org/standards/techs/components (visited on 07/24/2017).

[24] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. "Guidelines for Using Multiple Views in Information Visualization". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '00. Palermo, Italy: ACM, 2000, pp. 110–119. ISBN: 1-58113-252-2. DOI: 10.1145/345513.345271. URL: http://doi.acm.org/10.1145/345513.345271.

[25] Wikipedia. *Ember.js*. Sept. 20, 2017. URL: https://en.wikipedia.org/wiki/Ember.js (visited on 09/23/2017).

[26] J. S. Yi, Y. a. Kang, and J. Stasko. "Toward a Deeper Understanding of the Role of Interaction in Information Visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1224–1231. ISSN: 1077-2626. DOI: 10.1109/TVCG.2007.70515.