Geospatial Digital Dashboard for Exploratory Visual Analytics

Jonas Sjöbergh and Yuzuru Tanaka

Meme Media Lab, Hokkaido University, Japan
{js,tanaka}@meme.hokudai.ac.jp

Abstract. We present a system for visual data exploration, built using pluggable software components, which allows ad hoc combination of data from different sources ("data mash-up"). Interaction is done through "direct manipulation", making it easy to use for domain experts that may not be data mining or computer experts. All visualized results can be interacted with, and selections or groupings using one visualization result are automatically reflected in all other views of the same data.

1 Introduction

There are now many sources of "big data". Cheap storage and cheap sensors has made it possible to collect many types of data in huge quantities. With social networking, users in large numbers are also producing large amounts of user generated content. Examples of "big data" that it is now possible to get access to include: publicly collected data such as weather data, traffic records (accident records, traffic jam data), tracking data such as records of credit card usage, mobile phone location, car location (from the GPS navigation system), social networking data such as location tagged photos or short texts, and much more.

Combining different data sources can give insights not available using only one source, and different data sources can cover for each other when data becomes unavailable from one source.

We are currently involved in a project on using big data to help with disaster management, and in particular to help with snow removal in the city of Sapporo. Sapporo is a big city that gets very large amounts of snow every year, which is similar to a regularly occurring small natural disaster. We hope to improve the snow plowing and removing strategies by using "big data".

Big data is normally so big that it is difficult for a human to get a grip of the information in the data. Using data mining and statistical methods to extract the information we are interested in is one way to use big data. Sometimes, it is not obvious what information we want or how to extract the information we need from a set of data sources. Visualizing the data to a user can then help in getting ideas of what data to extract or how to process the data further.

When something out of the ordinary, like a natural disaster, happens, it is not always clear what data contains the information we want. The ideal data may also not be accessible to us at such times, and we may thus need to use other sources of data and do the best we can using these.

To help in situations such as when you do not know how to model your problem or when you have to access new unfamiliar data sources, we have developed an environment for visual exploration of data. It allows many types of visualizations of many types of data, and it allows ad hoc federation of different data sources (data mash-up) and adding of new visualization methods.

2 Snow Removal Project

In our project we are interested in using many sources of data to help dealing with snow in the winter here in Sapporo, and with disaster management in general. Sapporo is a city in northern Japan with 1.9 million citizens and an annual snowfall of about 6 meters. Plowing roads and removing snow to keep the city working during the winter is a big problem and costs about 147 million dollars per year. It is similar to a small natural disaster that occurs regularly, and can be used to try out disaster management strategies etc.

We have access to many sources of data, including:

– Probe car data, around 2000 taxis driving in Sapporo giving their time, location, and speed. We also have similar data from private cars, but the number of sensor enabled private cars is much lower.
– Traffic jam sensor data, from traffic jam sensors all over Sapporo.
– Multi-sensor weather data, measurements of wind, temperature, rain, snowfall, snow depth on the ground, moisture, etc. at 52 locations in and around the city. We also have weather radar data from a different source.
– Snow plowing and snow removal records. Data collected by hand by the companies that remove snow.
– Snow related call center complaints. Data collected by call center operators answering calls from citizens reporting snow related problems or complaining about the snow removal service.
– Subway data, the number of passengers entering and leaving each subway station of the three subway lines in the city.
– Traffic accident reports, collected by the police.
– Social networking data, time and location stamped texts from Sapporo from users of the Twitter social networking service.

One part of the project is building a library of tools to visualize/explore these data. A system for doing this is described in the next section. An older version of this system, as well as other things that are being done in the project, have been described elsewhere[1].

3 Digital Dashboard for Visual Data Exploration

The *Digital Dashboard* is a system for visual exploration of data. It allows visualization of data from many sources at the same time, using many types of visualization methods. The user interface is based on “direct manipulation”, and all visualization components are expected to allow manipulation of the visualization results. The direct manipulation can for instance be clicking and dragging on a map to select areas, clicking on representations of subsets of data in a clustering component, etc.

The *Digital Dashboard* is built using components. There are components for interfacing with data sources of various kinds, and components for visualizing data. The *Dashboard* provides a simple interface for components to connect to it (and expects all components to support a simple interface) and hides the components from each other. An individual component does not need to know anything about the data source components its data comes from or about other visualization components that are currently visualizing the same data, it only needs to interact with the *Dashboard*.

The *Dashboard* and its components are built using the *Webble World* framework[2], the latest version of the *Meme Media* (*IntelligentPad*) framework[3]. A *Webble* is a pluggable software component.

3.1 Webble Technology

The goal of *Meme Media* is to make functionality and services as easy to reuse as copy-pasting texts or images is today. *Meme Media Webbles* are intelligent pluggable software components that exist on the Web. The *Webble* technology allows wrapping existing software with interface wrappers and once some software has been wrapped it can be used together with other *Webbles*, and used/reused in many applications. *Webbles* can of course also be written from scratch.

The *Webble World* framework is currently implemented in Silverlight, and *Webble* software runs in any Web browser that has a Silverlight plugin. Currently, a new version of the *Webble World* framework is being built using HTML5 and Javascript instead of Silverlight, which will allow *Webbles* to run on many more platforms, such as mobile devices.

In the *Digital Dashboard* there are components written from scratch, components that wrap previously existing *Webbles* to conform to the interface expected by the *Dashboard*, and components that wrap previously existing external software. One example of the latter is a *Webble* that displays geospatial information on a map, which was built by wrapping the The ArcGIS[1] software with a *Webble* wrapping interface.

Webbles (and *IntelligentPads*) communicate using *slots*. A *slot* is an access point to a variable or method in the *Webble*, and *Webbles* that are connected to each other can have *slots* connected too. These *slots* will then exchange values whenever the value of one *slot* is changed, effectively making a variable a shared variable between the two *Webbles*.

[1] ESRI (2012) ArcGIS API for Silverlight (ver. 2.4) `http://help.arcgis.com/en/webapi/silverlight/`

3.2 Dashboard Internal Structure and Component Interface

The *Digital Dashboard* has three types of components, the *Digital Dashboard* parent component, visualization components, and data source components. New components can be built and plugged in at any time, even when the system is already running. As long as they conform to the simple interface expected by the *Dashboard* parent, they can be used as soon as they are plugged in.

Each component is plugged into the *Dashboard* parent, but the components have no knowledge of other components that may be running. All communication between components, such as sending the data from a data source component to a visualization component, goes through the *Dashboard*. Components can have (*Webble*) children of their own, as for example the *Map Interaction* component that acts as an adapter between the *Dashboard* and a previously existing *Webble* for using the ArcGIS framework.

All components plugged into the *Dashboard* are expected to have two *slots*: PluginName (specifying what this component should be called in menus etc.) and PluginType (specifying whether it is a visualization component or a data source component). The different types of components are then expected to have further *slots*, specific to the component type.

The data source components hide the underlying data storage format from the *Dashboard*. Data source components can for example be *Webbles* that allow access to an SQL database and return the data in the format expected by the *Dashboard*, *Webbles* that parse XML files, or *Webbles* that access Web services.

Data source components are expected to follow a simple interface. They are expected to have the following *slots*:

– ProvidedFormat, and XML string specifying what data this source provides. It details the data types of the different data fields, and what these fields should be called (in menus etc.).
– FormatChanged, signaling that the format of the data this source provides has changed.
– DataValuesChange, signaling that the data has changed and the parent needs to update any visualization components using data from this source.

They also have *slots* containing the actual data, one *slot* for each data field. The data is stored as vectors, so numerical data would be stored as a vector of doubles (or integers) and text data would be stored as a vector of strings, etc. These *slots* are described in the XML of the ProvidedFormat *slot*.

Visualization components are expected to have the following *slots*:

– ExpectedFormat, and XML describing what types of data the plugin expects and in which *slots* it expects to received the data.
– FormatChanged, signalling that the component has changed what types of data it expects.
– DataValuesSetFilled, as slot where the parent informs the plugin about which slots it has filled with data (some data fields could be optional, or several different sets of data could be allowed, etc.).

– DataValuesChanged, a slot where the parent signals that the data has changed and the plugin should parse the new data etc.
– LocalSelections, a slot where the plugin tells the parent which data items are selected/deselected/grouped together on this component.
– GlobalSelections, a slot where the parent tells the plugin the global selection status of the data items (e.g. a data item may be selected locally on one plugin but unselected on another component, making it unselected in total).
– GroupColors, information on what colors the plugins should use to visualize different groups of data (to get a uniform look across all components).

They also have *slots* for data input, similar to the *slots* for data output in the data source components, expecting vectors of strings etc. These are described in the XML of the ExpectedFormat *slot*.

In an older version of the *Digital Dashboard*, the component interface was slightly different. The older version used XML to pass data between data sources and components, while the current version uses vectors of primitive data types (or objects, for more complicated data types). The XML version had a cleaner interface, but generating and parsing XML for a lot of the communication turned out to be too slow when using the system on big data. A description of the previous interface can be found in [4].

Derived attributes can be computed at runtime and used for selecting subsets of data etc. One example is a clustering component that clusters data and then allows grouping and selecting data based on which cluster they belong to.

3.3 Usage Example

The *Digital Dashboard* is intended to work for visualizing and exploring any type of data, but since it is being developed in a project with focus on snow removal the first components developed are components that are helpful for data that may help with that. Since a lot of the information is geographic in nature, there is for instance a component that visualizes geographic data on a map. It could for instance be useful to show roads where the speed is currently lower than normal (perhaps caused by snow or ice) or locations of recent traffic accidents. A quick look could then give an indication of if there are areas that seem to have more problems than others, or if snow removal resources are currently limited, which areas to prioritize etc.

In Figure 1 an example of what the *Digital Dashboard* looks like is shown, showing one of the many setups possible. There are six visualization plugins and one data source plugin used in the example setup. There are also two other *Webbles* that are connected to one of the visualization components.

The data source component contains probe car data. The data is statistically treated data that originates from about 2,000 taxis driving around the city and reporting the time, location, and speed when they pass traffic lights. The data is then averaged for each road segment in the city, where a road segment is basically a stretch of road between two traffic lights or intersections. For each road segment the data contains statistical data for each five minutes period

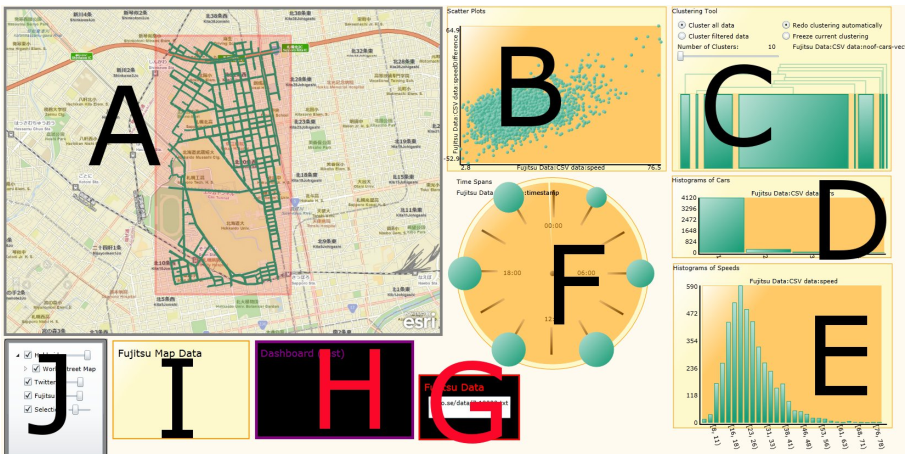

Fig. 1. The *Digital Dashboard* showing probe car data: time stamped traffic amount and speed readings for road segments. **A** is an ArcGIS map *Webble*, **B** is a scatter plot visualization component, **C** is a clustering component, **D** and **E** are two instances of a histogram component, **F** is a 24h clock component, **G** is the probe car data source component, **H** is the *Dashboard* parent *Webble*, **I** is a visualization component using the map *Webble* **A**, and **J** is a *Webble* to hide or make layers on the map transparent.

during each day. The data available for each segment and each five minutes period are: the minimum, maximum, and average speed; the time; the number of taxis (of the probe car taxis) with passengers in them that passed. The data also contains the location of the road segments, their lengths, etc.

The example setup shows a one day subset of the data from a winter day, and only from road segments around our university. The data is collected for every day, and all road segments in the city (over 100,000 road segments). In the example setup the data has been further averaged over four hour periods.

The visualization components used in the example setup are: a 24 hour clock to select time segments (**F**); two instances of a component showing histograms, here showing the average speeds and the average number of cars per five minutes (**D** and **E**); a scatter plot component, showing the average speed on the horizontal axis and the speed on this winter day compared to the average speed of the same road segment in the summer (the “winter speed down”) (**B**); a clustering component that shows clusters that the component created by clustering the road segments based on 288 dimensional (24 hours × 12 five minute periods per hour) vectors of the number of cars passing during the day (**C**); a map visualization component showing the road segments on top of a map of Sapporo (**I**). The last component uses an ArcGIS wrapper *Webble* to show maps (**A**), which in turn uses a *Webble* for controlling the layers on the map (**J**).

A simple example of interacting with the components is shown in Figure 2. In the upper image, component **D** was used to select only roads with a lot of taxi traffic. All other data items are automatically removed from all components

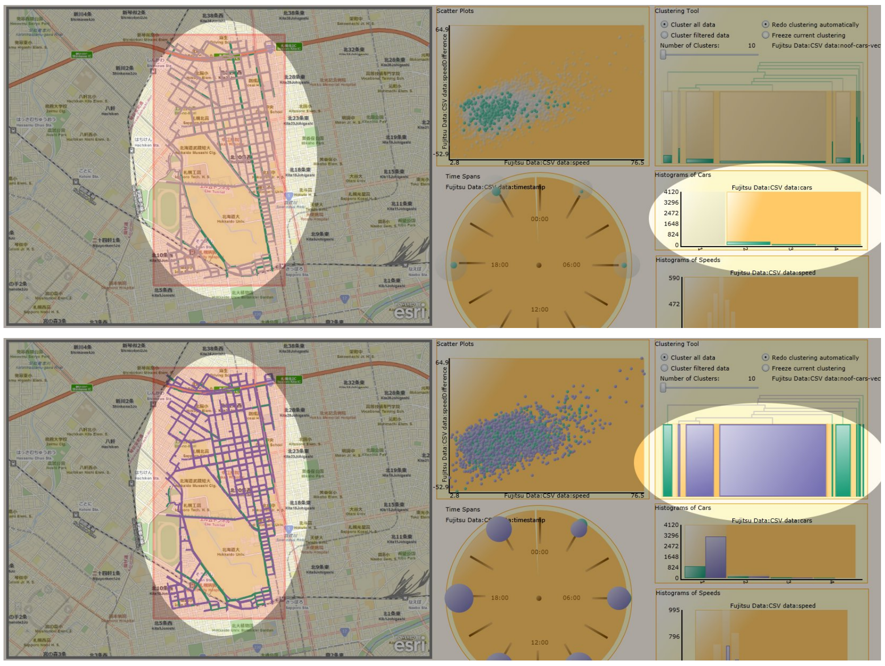
Scatter Plots
64.9
-52.9
2.8
Fujitsu Data:CSV data:speed
76.5
Fujitsu Data:CSV data:speedDifference
Clustering Tool
Cluster all data
Cluster filtered data
Number of Clusters: 10
Redo clustering automatically
Freeze current clustering
Fujitsu Data:CSV data:noof-cars-vec
Time Spans
Fujitsu Data:CSV data:timestamp
00:00
06:00
12:00
18:00
Histograms of Cars
Fujitsu Data:CSV data:cars
4120
3296
2472
1648
824
0
Histograms of Speeds
Fujitsu Data:CSV data:speed
590
472
995
796
esri

Fig. 2. Selecting road segments with a lot of traffic in two different ways.

visualizing the same data. The road segments still shown in green on the map correspond to the main roads around the university, as expected.

Data can also be divided into different groups and contrasted against each other. In the lower part of Figure 2, the clustering component (**C**) was used to divide the data into two groups based on the clustering result. Since the clustering was based on the amount of traffic throughout the day, the green group still corresponds well with the main roads around the university.

In Figure 3 the scatter plot has been used to select road segments with speed outliers, segments where the average speed was unusually high. The map has been zoomed in on a few of these to take a closer look. Some segments are stretches of the expressway through Sapporo, which naturally has a high average speed. Other road segments are mostly very short road segments with little traffic. A guess is that the high speeds are taxis rushing to pass a changing traffic light, which would give a large increase in the average speed if the segment is short and there are no other taxis there during the same time interval.

To find areas that may be in need of snow removal, we can select road segments that have a large speed difference compared to the average speed in the summer (at the same time of day and the same day of the week). A histogram component could be used to show such data and select road segments like that. In Figure 4 the scatter plot, which shows the speeds and the speed difference compared to the summer average speeds, is used to select road segments with

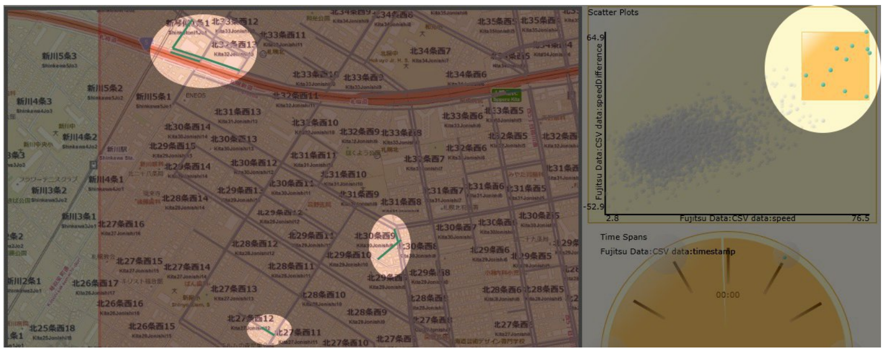
Scatter Plots
Fujitsu Data:CSV data:speedDifference
Fujitsu Data:CSV data:speed
Time Spans
Fujitsu Data:CSV data:timestamp
00:00

Fig. 3. A closer look at segments with speed outliers.

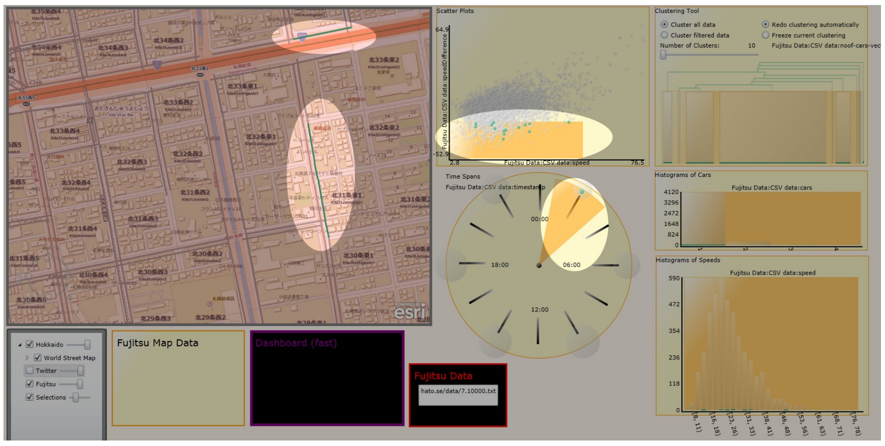

Fig. 4. A closer look at segments with potential snow or ice problems.

large speed downs. The 24 hour clock is also used to show only segments where the problems persist at night. If there are large speed downs even at night, when traffic is low, the cause is often ice making the road slippery or snow making the road narrow or in other ways difficult to use. The map has then been zoomed in on an area with several road segments with problems clustered together.

Since we do not know what caused the speed downs in the areas, it might be helpful to add data from a different source as a complement. In Figure 5, a data source component with data from the Twitter social networking service has been added. The data contains the text content, the time stamp, and the GPS location of a number of tweets from inside Sapporo on the same day as the probe care data already visualized.

One more instance of the map visualization component has also been added, to show the Twitter data on the map. In the upper image of Figure 5, the map has been zoomed out and we can see that most tweets are sent in the city center, as is to be expected since this is the area with the most people during the day.

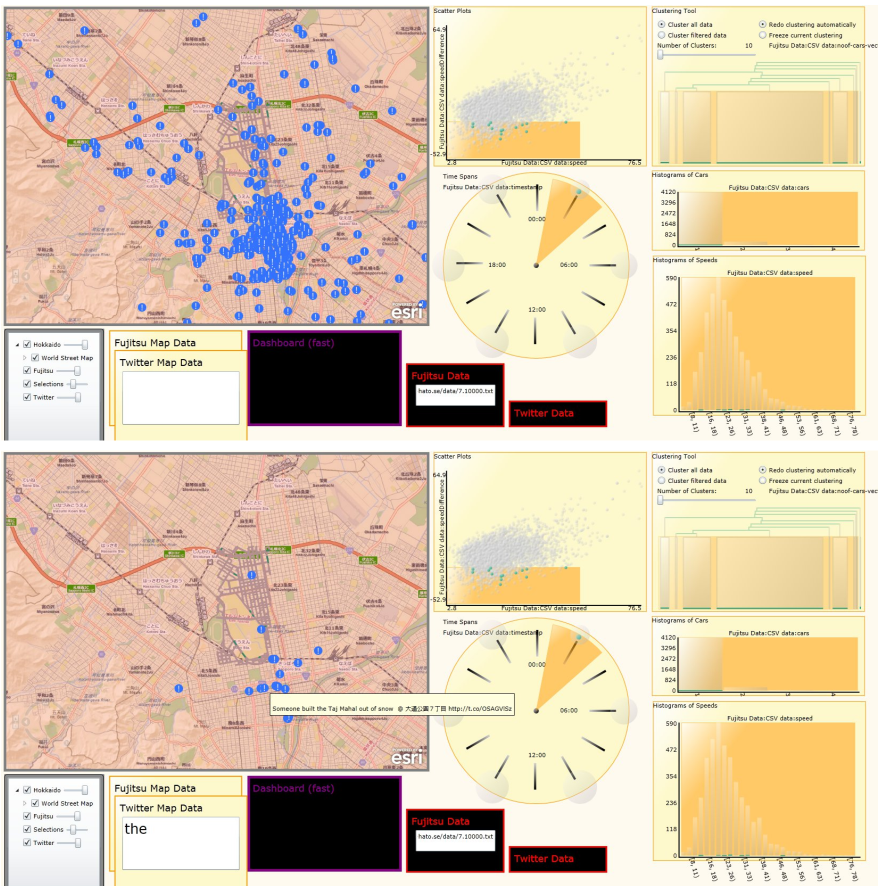
Someone built the Taj Mahal out of snow @ 大通公園７丁目 http://t.co/OSAGVlSz

Fig. 5. Adding data from Twitter, and filtering these data to show tweets in English.

In the lower image, the map visualization component has been used to select only tweets containing the word "the". This is a quick way to select tweets in English (most of which contain "the") and remove tweets in Japanese. Most Sapporo citizens write in Japanese on Twitter, and the remaining tweets are mainly tweets by tourists visiting the city. The tweets from the tourists are heavily concentrated to the city center, to the main street in Susukino and to the big Odori park in the middle of Sapporo. The day these data were collected was the first day of the Sapporo Snow Festival, which is located in the Odori park and draws huge amounts of tourists (around 2 million people per year) and of the Susukino Ice Festival, located in the Susukino area. The tourist tweets thus occur where you would expect them to occur, and checking the text content by hovering the mouse over a tweet we can see the Snow Festival related: “Someone built the Taj Mahal out of snow”.

We can check this data to see if someone is mentioning snow, ice, slipperiness, etc. near the road segments that seemed to have problems, but on this day there were no such tweets. The road segments around the festival areas also have huge speed downs during the day (when the festival area is open) and some road segments are even closed off during certain hours these days. If you were not already aware of the Snow Festival, checking the tweets in the area would quickly show you why these road segments have unusual data during these days. Adding other types of data from sources could also give more insights.

Apart from the simple usage examples shown above, all components can also be moved around (by clicking and dragging) to put visualization results you want together near each other. The components can be resized to see more details in a component you care more about and to waste less screen space on components that are not important for the moment. More components can be added at any time (as can other *Webbles*), and components can of course also be removed. The color scheme of the components and the colors used for the visualized data can also be changed easily through the *Dashboard* parent.

4 Similar Systems

Here we give a short overview of other systems for visualization and exploration of data, and point out some differences between them and our system.

RapidMiner[5] is an open-source prototyping system for knowledge discovery and data mining. It is widely used and supports very many data mining and machine learning algorithms. It is used to graphically set up work flows for data mining. It supports multiple views of the same data and you can go back and change some step of the work flow and get a different visualization results. Like in our system, the underlying data format is hidden from the data mining operators or visualization components, and changes made to a work flow are reflected in all views of those data. The interaction when setting up work flows is a visual process, but, there is no interaction with the visualized results.

Snap-Together Visualization[6] (*Snap*) is very similar to our system. Different visualization components are connected and e.g. selecting data in one component is automatically reflected in connected components. As in our system the interaction can be bidirectional so interactions with the second component are reflected in the first component too. Components can be connected in different ways, so selecting an item may select related items in one connected view, and open details about the selected item in another view. *Snap*, like our system, has a small interface that components need to follow and it is possible to wrap existing software with an interface wrapper (in for instance Visual Basic). Unlike our system, *Snap* expects the data to come from a database (supporting ODBC) and uses Microsoft’s COM for component (process) communication.

The *VERD*[7] system uses IntelligentBox, another version of *Meme Media* [3], for data visualization. It works with relational databases and has a wrapper to treat Web resources as relational schema too. Using direct manipulation, it allows interactive data exploration using various visualization methods. All visualization views set up are also treated as relations, making it possible to apply the same operations to visualization views as to raw data. *VERD* requires the IntelligentBox environment while our system runs in any Web browser, and in *VERD* the interaction only goes downstream in the visualization flow, i.e. interaction with one component does not affect components earlier in the flow.

DEVise[8] is a data exploration system for relational databases. It supports multiple views that can be connected so that e.g. zooming in one view is reflected in another view, and connections can be bidirectional. *DEVise* supports many types of operations on the data, but the types of user interaction with the visualization results are somewhat limited.

Tioga-2[9] (now called *Tioga DataSplash*) is a direct manipulation system for setting up visualizations of database contents. Multiple visualizations can be connected and changes in the range to visualize in one are then automatically reflected in others. *Tioga* has a fairly limited set of visualization primitives and only works with relational databases as the data source.

The *Trial Outline Builder*[10] (*TOB*) can be considered an older version of the *Digital Dashboard*. The *TOB* is a system for supporting clinical trials on cancer and can generate trial design plans (flow charts) and help with collecting data. It also has a data analysis part, built with components similar to the *Digital Dashboard* components, though they were written specifically for *TOB* and were not generic. They were built using *Webbles*, so they can easily be integrated into the *Dashboard*, though.

4.1 Available Components and Performance

The *Digital Dashboard* can be set up using any combination of the available components and the usage example in the previous section only shows one small example. New components are built when a need arises, so the number of available components is not fixed.

Currently the following data source components are available:

- XML Data Source, a data source component that parses XML files containing data and converts the data to the format expected by the *Dashboard* parent. Can be used together with existing *Webbles* that allow accessing Web Services to use Web Services providing data in XML format.
- CSV Data Source, a data source component that parses CSV (Comma Separated Vector) data files (used by many available data manipulating software, for example Excel) and converts the data to the format expected by the *Dashboard* parent.

There are also a number of visualization components available. Many instances of the same visualization component (or data source component) can be used at the same time, i.e. you can have several Bar Chart components, showing different data fields. Currently the following visualization components are available:

- Bar Charts, visualizes data using histograms or bar charts.
- Scatter Plots, visualizes two dimensional data as a scatter plot.
- Text Statistics, shows e.g. histograms of the most common words in text data.
- Clock, visualizes time data on a 12 or 24 hour clock.
- Point Data on Maps, visualizes geospatial data as points on maps (e.g. weather stations, or sent Twitter messages).
- Line Data on Maps, visualizes geospatial data as lines on maps (e.g. road segments).
- Trajectories on Maps, visualizes series of geospatial data on a map as points connected by lines (e.g. all Tweets by the same user ordered by time).
- Clustering, clusters vector data using various clustering algorithms, visualizes the clustering results, and allows selection or grouping of data items based on the clustering results.
- Life Tables, visualizes data as "Life Tables", survivability charts. Plots the number of data items (normally patients) that remain in a group (e.g. are still alive, or still relapse free) as a function of time (e.g. the number of weeks after being diagnosed). Frequently used to show or compare the efficacy of medical treatments.
- Storygraphs, visualizing geospatial data over time using Storygraphs [11]. These are visualizations where the horizontal axis is time, the left hand vertical axis is the latitude, and the right hand side vertical axis is the longitude. A data item is drawn on the line from its latitude to its longitude, at the point corresponding to its time.
- Parallel Coordinates, visualizes multi dimensional data using parallel coordinates [12], where coordinate axes are lined up beside each other and a point in the multi dimensional space becomes a series of line segments between the data axes.

All visualization components also allow interaction, so you can select or group data items by selecting bars in a bar chart, by selecting frequent words, by selecting areas on a map, by selecting time slices on a clock, etc.

Performance-wise, the more visualization components you use at the same time, the slower the visualization becomes. The main cost is that since there are more components that draw things, there are more things to draw and thus the time spent drawing on the screen is increased. The cost for the *Dashboard* parent coordinating more components is very low compared to the cost of drawing on the screen.

Some visualization components are much more performance intensive than others. It is for instance much more costly to draw dots representing each individual Twitter message on a map than it is to draw a histogram of how many messages were sent each day, since there are so many dots to draw on the map (the locations of the messages do not overlap very often, so the number of dots will be close to the number of messages) compared to drawing just a few rectangles for each day. Adding many more visualization components that do not draw that much on the screen does not impact the performance noticeably, while adding just a single component that is slow will have a large impact. Most components are written so as to only do incremental updates of the graphics (only redraw things that have changed), so once the initial visualization is done the system can be very fast as long as you do not do large changes in each step (e.g. deselect almost all data).

The amount of data that is being visualized also has a large impact on the performance, of course. The number of different data sources used does not have a large performance impact, but the total amount of data, or more precisely the amount of graphical elements that end up on screen, has a large impact.

The *Digital Dashboard* is still under development and has not been heavily optimized for performance. Currently, visualizing 4,000 traffic accidents using 10 different visualization components on a standard desktop computer is very responsive. All the visuals are updated in real time as soon as subsets of data are selected on any component or the data is grouped in new ways. Visualizing 4.5 megabytes of taxi probe car data using 10 different visualization components is fast (everything is immediately updated) if you make small changes (e.g. select small areas on the map) but when redrawing all visualizations (e.g. changing between selecting almost all data and almost no data) it takes a few seconds to update the graphics. Most of this time is spent redrawing the lines representing the road segments on the map.

5 Discussion

Using the *Meme Media* component based framework to build the system makes development and prototyping very fast. The *Dashboard* hiding the components from each other and only requiring a simple interface also makes it easy to develop new components. A wrapper to make existing *Webbles* work with the *Dashboard* can be written in a few hours, and writing a new component from scratch usually takes from one to a few days (depending on what the functionality of the component is).

The *Meme Media* also makes it easy to reuse software, and several of the components in our system have already been reused in other systems. We have incorporated components developed for other systems in the *Dashboard* too.

Making the *Dashboard* component interface generic and having all communication go through the *Dashboard* does make some things slower, and tightly integrating all components could make the system more efficient. A previous version of our system used XML as a cleaner and more generic way for components to communicate, but this turned out to be too slow to make the system interactive when visualizing big data. We believe the current design is a good compromise between run time efficiency and development speed and reusability.

Using direct manipulation as the main interaction method makes the system easy to use for users with little knowledge of computers or data mining. This means that users with domain knowledge (such as snow removal professionals in our case, or cancer specialists in another project we are involved in) can use the system and at the same time make use of their expertise in the field.

That all visualization results can be interacted with has been well received by users in the target audience (snow removal specialists, cancer researchers, etc.). It is powerful that you can use the visualization that tells you some subset of the data looks interesting directly to further refine your exploration.

Another thing that has been well received is that all views of the same data are automatically updated when selections or groupings in one view of the data are changed. It is possible to set up unconnected views of the same data too, if you want both types of behavior.

Derived attributes calculated from the data at run time can also be used like normal attributes in the system. Currently there is a clustering tool that clusters the data and you can then use the cluster ID together with other attributes to do further selections in the data. We are also building a pattern mining tool, other machine learning tools, and some statistical analysis tools, though these were not detailed in this paper.

6 Conclusions

We described a system based on pluggable software components. It allows ad hoc data exploration/visualization and ad hoc combination of data sources.

Interaction is done through "direct manipulation", which is intuitive and easy for non-experts (on data mining etc.) to use. All visualization results can also be interacted with, and selections or groupings based on one visualization result automatically update all other visualizations of the same data.

The component based technology used to build the system allows for fast prototyping, and it makes the ad hoc addition of new data sources or new visualization components easy.

References

1. Tanaka, Y., Sjöbergh, J., Moiseets, P., Kuwahara, M., Imura, H., Yoshida, T.: Geospatial visual analytics of traffic and weather data for better winter road management. In: Data Mining for Geoinformatics. (2013)
2. Kuwahara, M., Tanaka, Y.: Webble world – a Web-based knowledge federation framework for programmable and customizable Meme Media objects. In: The IET Intl. Conf. on Frontier Computing 2010, Taichung, Taiwan (2010) 372–377
3. Tanaka, Y.: Meme Media and Meme Market Architecture. IEEE Press, Piscataway; NJ; USA (2003)
4. Sjöbergh, J., Tanaka, Y.: Visual data exploration using Webbles. In: Proceedings of the Webble World Summit 2013. Volume 372 of Springer CCIS., Erfurt, Germany (2013) 119–128

5. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid prototyping for complex data mining tasks. In: KDD'06: Proceedings of the 12th ACM SIGKDD, Philadelphia, PA, USA (2006) 935–940
6. North, C., Shneiderman, B.: Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In: Proceedings of AVI'00, Palermo, Italy (2000) 128–135
7. Sugibuchi, T., Tanaka, Y.: Integrated visualization framework for relational databases and web resources. In: Proceedings of IHI'04, Dagstuhl Castle, Germany (2004) 159–174
8. Livny, M., Ramakrishnan, R., Beyer, K., Chen, G., Donjerkovic, D., Lawande, S., Myllymaki, J., Wenger, K.: DEVise: Integrated querying and visual exploration of large datasets. In: Proceedings of SIGMOD'97, Tucson, AZ, USA (1997) 301–312
9. Aiken, A., Chen, J., Stonebraker, M., Woodruff, A.: Tioga-2: a direct manipulation database visualization environment. In: Proceedings of ICDE'96, New Orleans, LA, USA (1996) 208–217
10. Sjöbergh, J., Kuwahara, M., Tanaka, Y.: Visualizing clinical trial data using pluggable components. In: Proceedings of the 16th International Conference on Information Visualisation IV'2012, Montpellier, France (2012) 291–296
11. Shrestha, A., Zhu, Y., Miller, B., Zhao, Y.: Storygraphs: Extracting patterns from spatio-temporal data. In: Proceedings of IDEA'13, Chicago, IL, USA (2013) 96–104
12. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multidimensional geometry. In: VIS'90: Proceedings of the 1st conference on Visualization '90, Los Alamitos, CA, USA (1990) 361–378